Graphene as a platform for novel nanoelectronic devices

Thesis by

Brian Standley

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2012

(Defended 14 May 2012)

To my parents

Acknowledgements

To join a nascent scientific movement shortly after its inception—in my case, the first transport experiments on a single graphene sheet by Novoselov et al. [1] in 2004—can be exhilarating, awe-inspiring, and, at times, deeply trying. The field of new endeavors seems endless, yet basic experimental practices are in flux or yet undiscovered. One climbs onto the shoulders of giants, yet cannot help but find the perch a bit wobbly and the view somewhat vertiginous.

Throughout this process I have been tirelessly encouraged by five individuals most of all: My adviser Marc Bockrath, himself a giant of carbon nanotube physics, has been a model of genuine scientific curiosity and integrity. My parents Larry Standley and Susan Blough have instilled in me a love of science and engineering and demonstrated how to work very, very hard without complaining (I am not sure how well the latter part of this lesson took). My brother Eric Standley has always been an earnest friend, in spite of all my teasing of him. Finally, my wife Eve Stenson has been a great adventure partner ever since we met (along with other positive attributes too numerous to mention here).

I enjoyed working with and getting to know many colleagues and friends at Caltech, including Anthony Mendez, Emma Schmidgall, Bertrand Bourlon, Henk Postma, Keisuki Shimizu, Cameron Hughes, Marcus Teague, Peter Hung, Hsin-Ying Chiu, Jinseong Heo, Vikram Deshpande, Hareem Maune, Isabella Kierk, and Sinchul Yeom. In particular, my (at least) weekly commutes to Riverside were made more enjoyable by the company (and driving services) of Sinchul, Anthony, and Isabella, and I must thank Hsin-Ying for hints on fabrication techniques.

My colleagues and friends at UC Riverside included Honest Makamba, Peng Wang, Tengfei Miao, Bin Cheng, Oleg Martynov, Juan Aguilera Servin, Cheng Pan, Yong Wu, Hang Zhang, Wenzhong Bao, and Jairo Valesco Jr. I very much appreciate how the Bockrath group in Riverside provided a positive environment in which to finish my projects and how the Lau group generously

offered assistance and advice (not to mention short-term loans of precious liquid helium). Special thanks are owed to Hang, Peng, and Tengfei for their enthusiasm (and patience) for debugging and improving *Mezurit 2*.

In addition to encouragement from and collaboration with outstanding people, much support is required to complete an experimental PhD. Faculty members Julia Greer, Shuki Bruck, Sandra Troian, Keith Schwab, and Nai-Chang Yeh provided valuable mentoring while Christy Jenstad and Rosalie Rowe helped in ways beyond normal administrative duties. I also wish to thank Alireza Ghaffari for maintaining the Watson cleanroom and Craig Graham for quickly fixing the superconducting magnet power supply.

My life in Southern California was greatly enriched by getting to know my housemates (and family-away-from-home) Matthew Kelley, Neil Halelamian, and Jessie Rosenberg, along with the multitudes of potluck dinner companions and ballroom dancers, who were uniformly warm and welcoming. Finally, I am compelled to mention Xerion (the cat), who provided many hours of lap-warming during late nights spent writing and programming.

Brian Standley

Pasadena, California

May 2012

Abstract

Graphene's superlative electrical and mechanical properties, combined with its compatibility with existing planar silicon-based technology, make it an attractive platform for novel nanoelectronic devices. The development of two such devices is reported—a nonvolatile memory element exploiting the nanoscale graphene edge and a field-effect transistor using graphene for both the conducting channel and, in oxidized form, the gate dielectric. These experiments were enabled by custom software written to fully utilize both instrument-based and computer-based data acquisition hardware and provide a simple measurement automation system.

Graphene break junctions were studied and found to exhibit switching behavior in response to an electric field. This switching allows the devices to act as nonvolatile memory elements which have demonstrated thousands of writing cycles and long retention times. A model for device operation is proposed based on the formation and breaking of carbon-atom chains that bridge the junctions. Information storage was demonstrated using the concept of rank coding, in which information is stored in the relative conductance of multiple graphene switches in a memory cell.

The high mobility and two dimensional nature of graphene make it an attractive material for field-effect transistors. Another ultrathin layered material—graphene's insulating analogue, graphite oxide—was studied as an alternative to bulk gate dielectric materials such as Al_2O_3 or HfO_2. Transistors were fabricated comprising single or bilayer graphene channels, graphite oxide gate insulators, and metal top-gates. Electron transport measurements reveal minimal leakage through the graphite oxide at room temperature. Its breakdown electric field was found to be comparable to SiO_2, typically $\sim$1–3 $\times$ 10^8 V/m, while its dielectric constant is slightly higher, $\kappa \approx 4.3$.

As nanoelectronics experiments and their associated instrumentation continue to grow in complexity the need for powerful data acquisition software has only increased. This role has traditionally been filled by semiconductor parameter analyzers or desktop computers running *LabVIEW*. *Mezurit 2* represents a hybrid approach, providing basic *virtual instruments* which can be controlled in concert through a comprehensive scripting interface. Each virtual instrument's model of operation is described and an architectural overview is provided.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graphene, a one-atom-thick layer of carbon arranged in a honeycomb lattice, has a long history as a mathematical construct but a short one as a physical reality. Its parent material, graphite, is naturally occurring and even found use in the pottery industry during the Neolithic Age [2]. Graphene, by contrast, was not isolated in a transport experiment-compatible way until the year 2004 using a simple but clever technique involving adhesive tape [1]. The electronic band theory of both "single layer" and bulk graphite was developed in 1947 [3], effectively kicking-off the era of graphene. Between these milestone dates, graphene was primarily used as the conceptual basis for lower-dimensional carbon materials such as nanotubes and C_{60} (which can be thought of as rolled or folded graphene sheets, respectively), although it was in fact isolated via reduced graphite oxide and imaged by transmission electron microscopy in 1961 [4].

The distinction between the graphene as a "platonic form" and the alternative—an actual sample[1]—is worth noting because it mirrors the challenges and opportunities of research. Graphene's superlative mechanical and electrical properties stem from its strong covalent C–C bonds and unique electronic band structure, which provides a wealth of new physics [5]. Access to the field, however, requires a real sheet of graphene complete with rough edges and a huge surface-to-volume ratio, making it highly sensitive to its environment. These challenges are, in fact, also opportunities but in disguise: The chemistry of the graphene edge is itself an emerging field and the surface of graphene can be functionalized to make sensors and other devices. Work directed toward these practical concerns advances the more fundamental side of the field as well, providing a more-complete platform for mesoscopic physics.

[1] Never is the contrast more stark than for a new student learning to deposit graphene in the lab, to whom it may seem that the idea of graphene is indeed all that exists.

This thesis reports on work that demonstrates both the utility of graphene for applications—a graphene-based nonvolatile memory technology in chapter 3 and a graphite-graphite oxide field-effect transistor in chapter 4—and as a building block for future physics experiments in the form of the graphene nanogap (also in chapter 3) and the graphite oxide sheet, which is a versatile nanoscale insulating fabric (again, chapter 4).

In addition, there are two chapters devoted to engineering concerns. Chapter 2 covers fabrication and measurement techniques for graphene-based devices. These measurement techniques involved at times complex collections of instrumentation, which were managed and controlled by a custom free software[2] application called *Mezurit 2*. Chapter 5 describes the context in which the application was conceived, its features and their models of operation, and its internal architecture. Though not all of this information would be essential to replicate this work, it is hoped that the additional exposition may help others who wish to extend *Mezurit 2* or create similar software. In many cases, the scientist who conceives of an experiment is most qualified to write the associated software yet may not have a computer programming background. Thus, an account of the internal workings of such a program written from a scientist's perspective would seem to be uniquely valuable.

The rest of *this* chapter is devoted to the electronic properties of graphene, specifically its band structure and density of states, along with the quantum correction to its capacitance. The conductance quantum is also discussed, as it appears in many areas of mesoscopic physics, including graphene-based nanoelectronics.

1.1 Electronic properties of graphene

1.1.1 Band structure

Figure 1.1a shows the real-space triangular lattice with a basis of two carbon atoms that forms graphene. The lattice vectors are

$$\vec{a}_{[1,2]} = a\left(\frac{\sqrt{3}}{2}\hat{x} \pm \frac{1}{2}\hat{y}\right) \tag{1.1}$$

[2] Here, free software is used to mean that not only is the source code available, but that users' freedoms are fully respected as defined by the GNU General Public License [6].

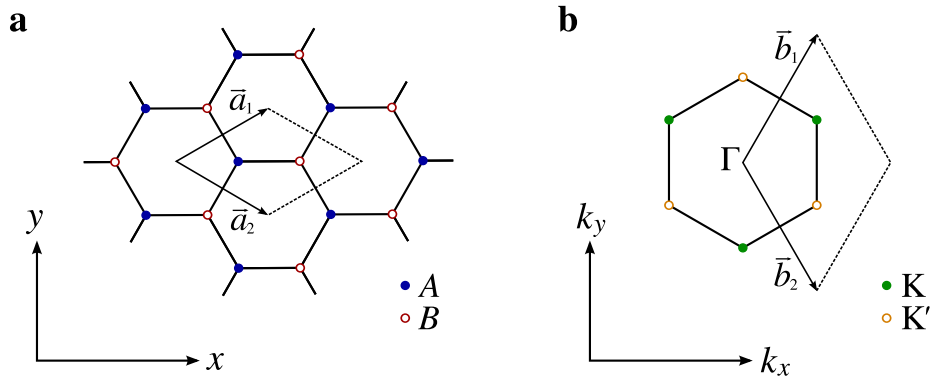


Figure 1.1: Graphene lattice in real and reciprocal space. (a) Real space triangular lattice with constant a = 2.46 Å containing two basis atoms per unit cell. (b) Reciprocal lattice vectors with hexagonal Brillouin zone and symmetry points Γ, K, and K′. (Adapted from Castro Neto et al. [7].)

with lattice constant a = 2.46 Å. The reciprocal lattice vectors $\vec{b}_{[1,2]}$ may be found by solving $\vec{a}_i \cdot \vec{b}_j = 2\pi\delta_{ij}$:

$$\vec{b}_{[1,2]} = \frac{4\pi}{\sqrt{3}a}\left(\frac{1}{2}\hat{x} \pm \frac{\sqrt{3}}{2}\hat{y}\right) \tag{1.2}$$

The hexagonal Brillouin zone, shown in Figure 1.1b, has two inequivalent corners K and K′ located relative to Γ at

$$\overrightarrow{\Gamma\mathrm{K}}, \overrightarrow{\Gamma\mathrm{K}'} = \frac{2\pi}{\sqrt{3}a}\hat{x} \pm \frac{2\pi}{3a}\hat{y}. \tag{1.3}$$

Three of the four valance electrons of each carbon atom participate in the covalent sp^2 C–C bonds, while the fourth is free to roam through the $2p_z$ orbitals [8]. The band structure of graphene may be calculated using a tight-binding approach, following Wallace [3] and Castro Neto et al. [7]. Considering only nearest-neighbor hopping, i.e., *A* sites to *B* sites and vice versa, produces the dispersion relation

$$E(\vec{k}) = \pm t\sqrt{1 + 4\cos\left(\frac{\sqrt{3}k_x a}{2}\right)\cos\left(\frac{k_y a}{2}\right) + 4\cos^2\left(\frac{k_y a}{2}\right)} \tag{1.4}$$

where parameter $t \approx 2.5$ eV [9] is the hopping energy. It is plotted for the region of k-space encompassing the Brillouin zone in Figure 1.2. The upper surface represents electron states and the lower surface, hole states. They meet at the six corners of the Brillouin zone called Dirac points.

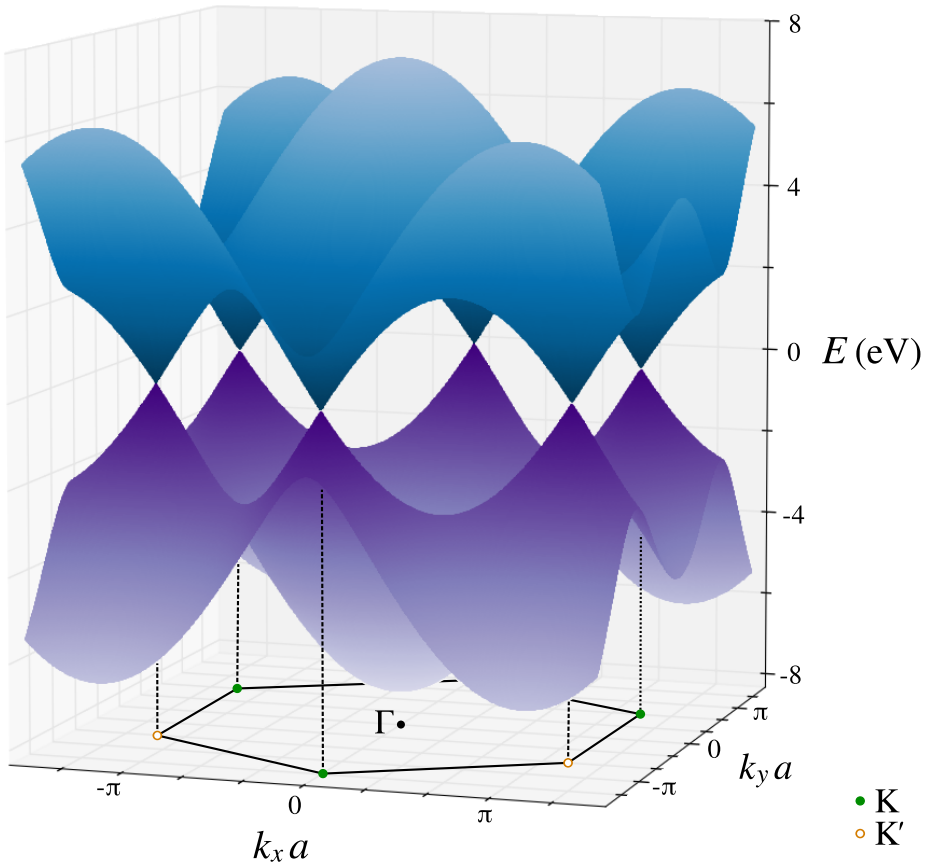


Figure 1.2: Energy as a function of $\vec{k}a$ for electron states (upper blue surface) and hole states (lower purple surface) in graphene. The two meet at the six Dirac points, labeled K or K′.

A low-energy approximation may be made by expanding equation 1.4 around K or K′ which yields[3]

$$E\left(\overrightarrow{\Gamma \mathrm{K}} + \vec{q}\right) \approx \pm t\sqrt{\frac{3}{4}q_x^2 a^2 + \frac{3}{4}q_y^2 a^2} = \pm t\frac{\sqrt{3}a}{2}|\vec{q}\,| \tag{1.5}$$

with group velocity $\vec{v} = \hbar^{-1}\vec{\nabla}_q E(\vec{q}) = \pm\hbar^{-1}t\sqrt{3}a/2\,\hat{q}$. Thus, the low-energy dispersion is

$$E(\vec{q}) = \pm\hbar v_F|\vec{q}\,| \tag{1.6}$$

where $\hbar$ is Plank's constant divided by 2π and $v_F = \hbar^{-1}t\sqrt{3}a/2$ is the Fermi velocity, approximately 8×10^5 m/s. Note that, unlike in systems with a free-electron-like parabolic dispersion relation, v_F does not depend on energy. $E(\vec{q})$ is plotted on top of $E(\vec{k})$ along KK′ in Figure 1.3.

[3] Only terms under the radical with a total order in q_x and $q_y \leqslant 2$ are kept.

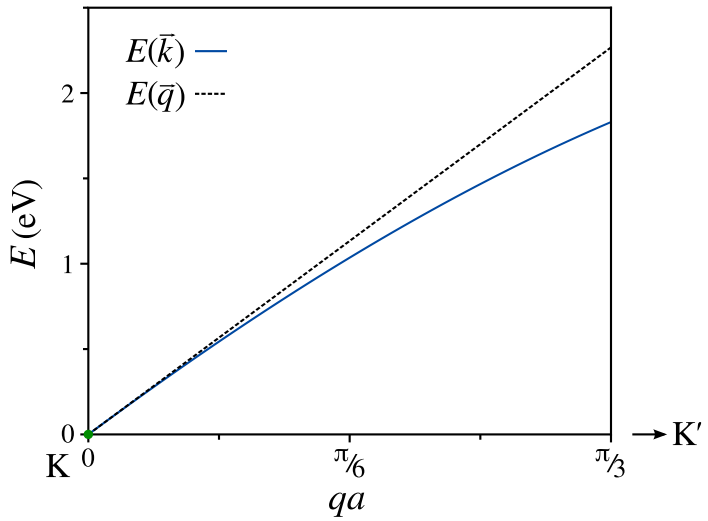


Figure 1.3: Low-energy dispersion $E(\vec{q})$ (equation 1.6) plotted along KK′ with $E(\vec{k})$ (equation 1.4) for comparison

1.1.2 Density of states

The density of states near the Dirac points may be calculated by adding up the states with energy $\leqslant E$. Given the rotational symmetry of $E(\vec{q})$ the available area of k-space is

$$A_k(E) = \pi \left(\frac{E}{\hbar v_F} \right)^2 . \tag{1.7}$$

Each k-state "occupies" $(2\pi/L)^2$ of k-space (where L is the length of the sample) and has fourfold degeneracy due to spin and the two inequivalent Dirac points, so the number of states is

$$N(E) = A_k(E) \frac{4}{(2\pi/L)^2} = \frac{L^2}{\pi} \left(\frac{E}{\hbar v_F} \right)^2 . \tag{1.8}$$

Thus, the density of states per area of graphene is

$$D(E) = \frac{1}{L^2} \frac{dN}{dE} = \frac{2|E|}{\pi \hbar^2 v_F^2} . \tag{1.9}$$

Interestingly, the experimental minimal conductivity of graphene is finite and typically on the order of $4e^2/h$ [10] even though $D(E_F)$ vanishes when the Fermi level E_F is tuned exactly to the Dirac point. One possible explanation is the presence of electron and hole "puddles", caused by disorder, occurring even in sheets with zero average doping [11].

1.1.3 Quantum capacitance

The commonly used expression for charge density in graphene placed near a gate electrode at voltage V_g with geometrical capacitance C_g' per unit area is

$$ne = C_g' V_g \tag{1.10}$$

which neglects the shift in Fermi level necessitated by graphene's finite density of states. Taking this into account, the density is $n = \int_0^\infty D(E) f(E_F, T) dE$ where $f(E_F, T)$ is Fermi-Dirac distribution at temperature T. At T = 0, the density is

$$n = \frac{2}{\pi \hbar^2 v_F^2} \int_0^{E_F} E dE = \frac{1}{\pi} \left(\frac{E_F}{\hbar v_F} \right)^2 \tag{1.11}$$

meaning that

$$E_F(n) = \hbar v_F \sqrt{\pi n}. \tag{1.12}$$

As an aside, equation 1.12 may be used calculate the shift in Fermi level for a known electron or hole concentration. Consider a graphene device equipped with a gate separated from the sheet by 300 nm of SiO_2 (ϵ_{ox} = 3.9 ϵ_0). Applying the equation 1.10 with V_g = 14 V yields $n = 10^{12}$ cm^{-2} and E_F = 93 meV. Comparing this energy scale to the region of overlapping curves in Figure 1.3 lends support to the efficacy of the low-energy approximation.

The charge density may be determined in a self-consistent manner by separating the electrostatic and kinetic energies such that

$$eV_g = e\phi + E_F \tag{1.13}$$

where ϕ is the electrostatic potential of an electron on the graphene sheet. Substituting in equation 1.12 and $\phi = ne/C_g'$ and solving yields an improved version of equation 1.10 [12]:

$$ne = C_g' V_g - n_q e \left(\sqrt{1 + 2\frac{C_g' V_g}{n_q e}} - 1 \right) \quad \Big| \quad n_q e = \frac{\pi (\hbar v_F C_g')^2}{2e^3} \tag{1.14}$$

Using this equation for the example above results in a 0.7% decrease in charge due to this quantum correction to the capacitance.

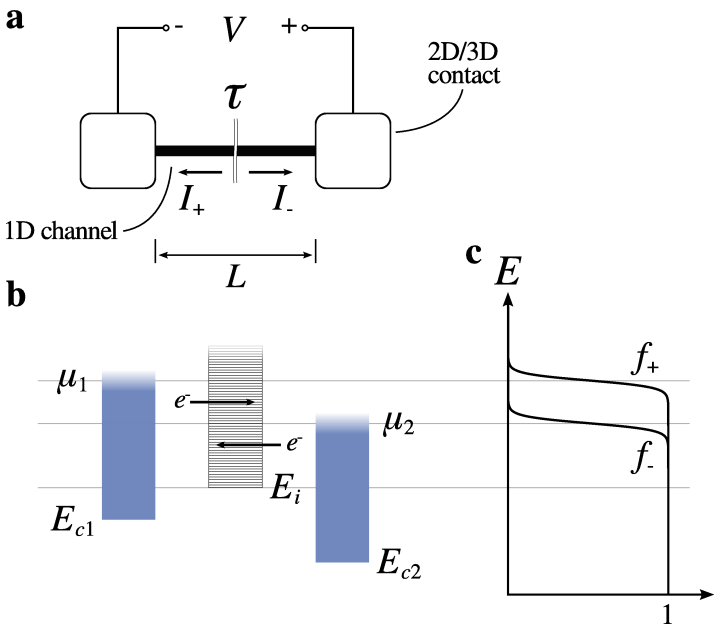


Figure 1.4: Schematic and band diagram of a 1D channel between 2D/3D contacts. (a) Circuit schematic showing virtual currents I_+ and I_-. The channel includes a transmission coefficient τ. (b) Band diagram of contact conduction bands and channel states. The right-moving electrons contribute to I_+ and left-movers, to I_-. (c) Occupation functions for electrons injected into the channel. (Adapted from Datta [16].)

1.2 Conductance of a point contact

The conductance quantum $G_0 = 2e^2/h$, where e is the electron charge and h is Plank's constant, is common in mesoscopic physics such as the study of point contacts or one-dimensional (1D) conductors. Examples of the former include metallic break junctions [13] and gated constrictions in two-dimensional electron gases [14]. The latter include carbon nanotubes which support ballistic conduction over μm length scales and have different 1D sub-bands depending how they are "rolled-up". Carbon nanotubes are not the only known carbon-based 1D conductors, however, given evidence for the existence of carbon-atom wires reported by Rinzler et al. in 1995 [15]. The quantized conductance of a single, spin-degenerate channel is derived below.

Figure 1.4a shows a model system comprising a channel between bulk (2D or 3D) contacts, to which a bias voltage V is applied. The energy of an electron traversing a one-dimensional channel of length L is

$$E = E(k) + E_i \tag{1.15}$$

where $E(k)$ is the dispersion relation and E_i is the band edge, which includes the additional energy arising from the lateral confinement of a particular mode i. Assuming the channel is a ballistic conductor save for a transmission coefficient $\tau \leqslant 1$, the right-moving electrons produce a virtual current

$$I_+ = \sum_{+k} nevf_+\tau \tag{1.16}$$

where $ne = 2e/L$ is the electron density per k-state (including spin degeneracy), $v = \hbar^{-1}\, dE/dk$ is the group velocity, and f_+ is the occupation function given the known chemical potential μ_1 of the left-hand contact. A simple band diagram is shown in Figure 1.4b. Integrating over k-space yields:

$$I_+ = \frac{L}{2\pi}\int_0^\infty \frac{2}{L} e \frac{1}{\hbar}\frac{dE}{dk} f_+ \tau dk = \frac{2e}{h}\int_{E_i}^\infty f_+ \tau dE$$

Subtracting the equivalent expression for the left-moving electrons and dividing by V gives the conductance:

$$G = \frac{I_+ - I_-}{V} = \frac{2e}{h}\frac{e}{\mu_1 - \mu_2}\int_{E_i}^\infty (f_+ - f_-)\,\tau dE \tag{1.17}$$

Assuming τ is independent of energy and $eV \ll k_B T$ (f_+ and f_- are shown for nonzero temperature in Figure 1.4c), the integral in equation 1.17 becomes $(\mu_1 - \mu_2)$, giving a linear response conductance

$$G = \frac{2e^2}{h}\tau. \tag{1.18}$$

Note that this predicts a finite conductance for even highly transmitting channels resulting from the interface with the contacts [16]. This quantity is the conductance quantum $G_0 \approx 77.5\,\mu\text{S}$.

Carbon-atom wires are predicted to have a conductance which oscillates from $\approx$1–2 G_Q with the number of atoms in the chain [17]. These chains are used to model the conductance of graphene nanogaps in chapter 3.

Chapter 2

Fabrication and measurement techniques

2.1 Introduction

The graphene-based devices for this work were made using a combination of standard and proprietary techniques. The basic formula included random deposition of graphene and/or graphite oxide onto silicon (covered by a layer of SiO_2) substrates with predefined grids of alignment marks. Carefully designed source, drain, and, sometimes, top-gate electrodes were then created using electron-beam lithography (EBL) and either thermal or electron-beam evaporation of metal followed by liftoff. These basic steps are described in detail along with alternative approaches to provide context and directions for future improvements.

2.2 Graphene deposition

The practical process of transferring a single graphene sheet from a bulk graphite source to a substrate is quite simple considering the late date (2004) at which is was first definitively demonstrated [1]. Several early methods are summarized:

Microcantelever cleavage Arrays of μm-scale square graphite pillars are made at the surface of highly-oriented pyrolytic graphite (HOPG) by etching the surrounding material. Individual pillars are then detached and affixed to silicon microcantelevers using a precision micromanipulator. Finally, an atomic force microscope (AFM) is used to drag the pillars across a substrate, leaving a trail of thin[1] graphite layers [18].

[1] This method was demonstrated to produce graphite layers as thin as 10 nm.

Mechanical exfoliation Large (20 µm–2 mm) graphite mesas prepared from HOPG via oxygen plasma etching are immobilized in photoresist. The mesas are then repeatedly thinned with adhesive tape before being released into an acetone suspension which is applied to a substrate [1].

Drawing Adhesive tape is used to cleave HOPG or Kish graphite,[2] exposing a fresh surface which is dragged across a substrate by hand. Occasionally, graphene's strong interaction with the SiO_2 surface [20, 21] causes sheets to be "pulled out" of the bulk [22].

Both the drawing method and a variation of the mechanical exfoliation method (sometimes called the modified Scotch tape method) were used in this work. The latter seems to be the most commonly used method at the present time: Adhesive tape removes a thick layer of graphite from the bulk, which is repeatedly cleaved by folding and unfolding the tape. The tape, now resembling a small, metallic-grey Rorschach test, is then placed directly on the substrate and slowly peeled off. With practice this method can reliably produce graphene sheets, although residual adhesive is a concern [23].

The drawing method, while cleaner than the modified Scotch tape method, can be finicky in practice. Optimizing factors such as cleaving rate, pressure, and drawing speed can be challenging and at times the process seems to be more art than science. There are several ways to promote the transfer process, such as electrostatic deposition, in which a bias is applied between the substrate and graphite source [24], or starting with expanded graphite [25]. Other improvements were explored such as thinning flakes on-chip by pressing and peeling PDMS (polydimethylsiloxane), sometimes leaving single sheets underneath. Low-power oxygen plasma was found to remove the top layer from multilayer sheets, although the process was not found to be controllable and may introduce defects.

A single- or bilayer graphene sheet deposited using the drawing method is shown in Figure 2.1a. The sub-nanometer-thick graphene is visible in spite of its ≈98% light transmittance [26] due to the change in interference color of the 290 nm oxide layer [27] used throughout this work. The physical thickness of a graphene sheet may be measured by AFM. An example image showing both single- and multilayer regions is shown in Figure 2.1b. Figure 2.1c shows a histogram of the height as measured over a small region containing both bare substrate and graphene. The peak-to-peak

[2] Kish graphite was originally a by-product of steel production [19].

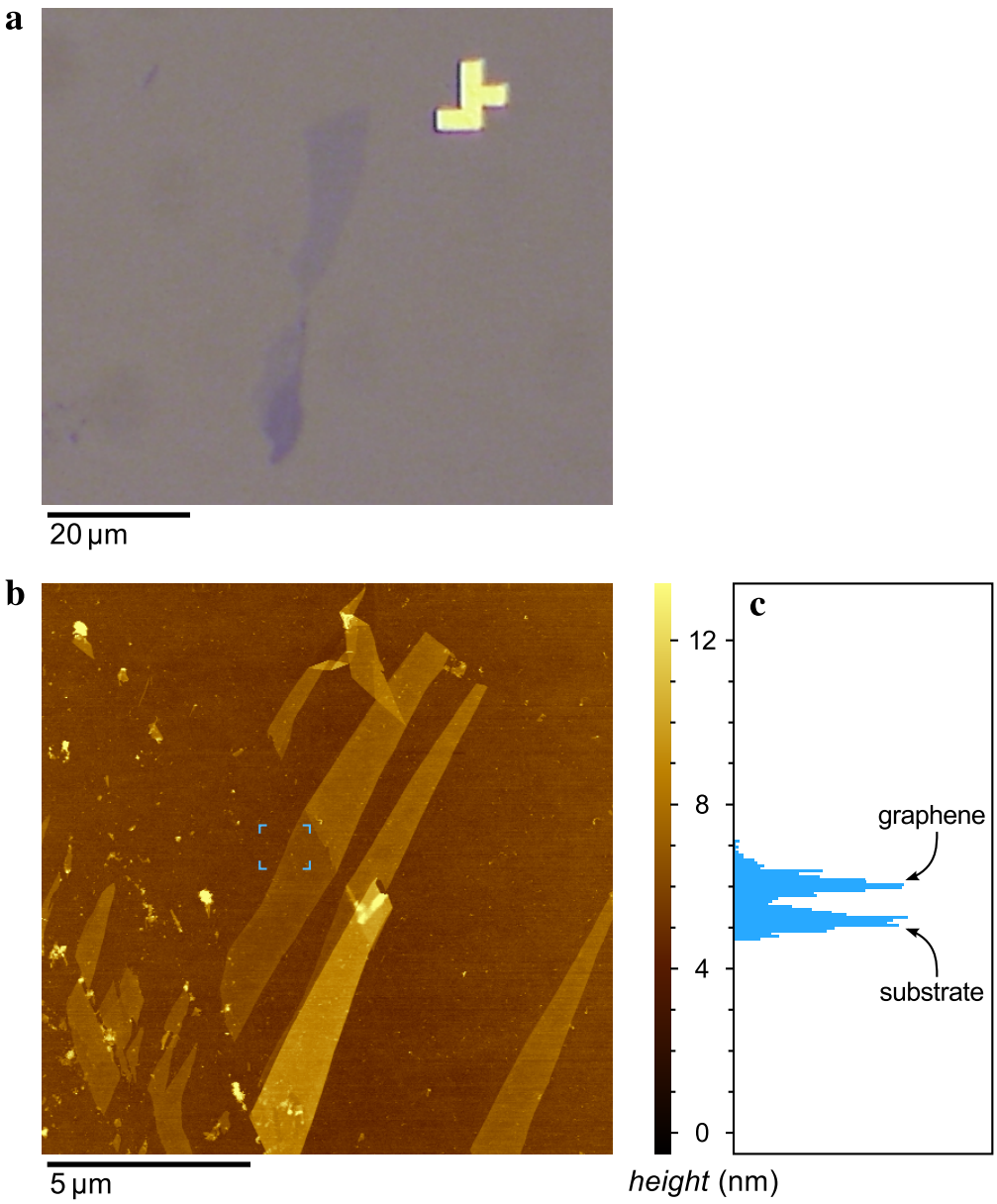


Figure 2.1: Micrographs of graphene on SiO_2 substrate. (a) Optical micrograph of graphene near a metal alignment mark. (b) False color atomic force micrograph of graphene, with height scale bar at right. (c) Height histogram (with arbitrary units) generated from the area indicated by blue marks in (b). The peak spacing, a measure of the thickness of the sheet, is 0.9 nm.

spacing is 0.9 nm, suggesting that the graphene is a single layer. However, the measured substrate-to-graphene step height may vary with AFM tip sharpness and environmental conditions, making AFM an unreliable method of layer-counting. Indeed, measurements by two different tips on the *same sheet* (not shown) produced values of 0.7 nm and 1.8 nm.

Recently, wafer-scale graphene sheets have been produced by chemical vapor deposition (CVD)

on nickel [28, 29] or copper [30]. While the carrier mobility of CVD graphene has improved, the traditional methods described above are still used for many experiments. Ironically, even some mobility-independent experiments benefit from starting with smaller sheets because no etching is then required to make space for electrodes.

2.3 Graphite oxidation

Graphite oxide (GO)[3] is graphene's native oxide in the same sense that SiO_2 goes with silicon. The GO used in this work was made by oxidizing synthetic graphite powder following the method of Hummers and Offeman [32] first reported in 1958. The process was scaled down and is summarized in Table 2.1. Temperature control proved challenging in practice (particularly during one nearly-disastrous early attempt), but nonetheless the resulting GO flakes had properties similar to those reported elsewhere [31, 33].

Table 2.1: Summary of modified Hummers method for oxidizing graphite

Step	Process	Amount
1	Heat sulfuric acid to 66 °C.	46 mL H_2SO_4
2	Transfer to an beaker surrounded by an ice bath and stir in synthetic graphite powder[a] and sodium nitrate.	2 g graphite, 1 g $NaNO_3$
3	Slowly add potassium permanganate while stirring so as to keep the temperature below 20 °C.	6 g $KMnO_4$
4	Replace the ice bath with a water bath on a hot plate (for added thermal mass) and maintain temperature at 35 °C for 30 min to 3 h.	
5	Slowly stir water, causing bubbling and a rise in temperature to 95 °C.	92 mL
6	Maintain temperature at 95 °C for 15 min.	
7	Further dilute with water.	280 mL
8	Add hydrogen peroxide, turning solution bright yellow.	5 mL H_2O_2 (30%)
9	Filter and wash GO solids.	

[a] Sigma-Aldrich, <20 µm

[3] The term graphite oxide is used throughout this work because most of the layers used were significantly thicker than the 1 nm reported for a single sheet by Stankovich et al. [31]. (Graphene oxide is indeed the appropriate term for a single oxidized sheet.)

2.4 Processing

Fabricating graphene-based devices using random deposition normally requires a set of custom metallic electrodes for each sample, although there are exceptions. For example, careful application of drawing method across a trench combined with predefined electrodes can produce a graphene device with no additional lithography [34]. However, the common method of using electron-beam lithography to define electrodes in reference to an alignment grid was used throughout this work:

1. Create a metallic alignment grid encompassing the majority of a $\sim$0.5 cm Si/SiO_2 substrate.
2. Deposit graphene sheets using the drawing or modified Scotch tape method.
3. Locate one-to-few-layer sheets using optical microscopy and record reference micrographs.
4. Design a custom electrode pattern based on the locations of the sheets relative to the alignment grid.
5. Deposit metallic electrodes using electron-beam lithography and thermal or electron-beam evaporation followed by liftoff.

Steps one and two are sometimes reversed such that a only a small alignment grid is required given known regions of graphene sheets. Fortunately, a wafer-scale electron-beam pattern generator (Leica EBPG 5000) was available in the Kavli Nanoscience Institute cleanroom at Caltech, allowing many identical large-scale grids to be made at once. Thus, the fabrication process for each sample really begins at step two, meaning that only a single additional lithography step is required. This lithography was most often run on a Hitachi S-4100 field-emission scanning electron microscope (SEM) located in Caltech's Micro/Nano Fabrication Laboratory.

2.4.1 Electron-beam lithography

Electron-beam lithography using a converted scanning electron microscope offers rapid prototyping of devices with scales from 100s of μm down to $\sim$50 nm with a minimum of substrate and pattern preparation. The standard recipe used in this work is listed in Table 2.2. The procedure is surprisingly tolerant of variation in its parameters with the exception of the electron exposure dose of step six.

Polymethylmethacrylate (PMMA) masks created with EBL are then used to metallize areas of the substrate. Thermal or electron-beam evaporation is used to deposit a uniform metal film

Table 2.2: Typical electron-beam lithography process flow

Step	Process	Parameters
1	Heat substrate on hotplate to remove adsorbed water.	115 °C, 5 min
2	Spin on first layer of resist.[a]	4000 r/min, 45 s
3	Briefly prebake to set first layer of resist.	170 °C, 5 min
4	Spin on second layer of resist.[a]	4000 r/min, 45 s
5	Finish prebaking.	170 °C, 60 min
6	Expose pattern with electron beam.	30 keV, 300 μC/cm^2
7	Develop pattern in 1:3 MIBK:IPA.[b]	50–60 s

[a] MicroChem NANO, first layer: MAA EL9, second layer: PMMA 950 C2.
[b] Methyl isobutyl ketone : isopropyl alcohol

across the entire substrate. Typically, the metal film comprises 40 nm of gold over a relatively thin chromium or titanium underlayer to aid adhesion. The PMMA is then dissolved in acetone to "lift off" the excess metal leaving only the metal that was in contact with the sample. The EBL, metallization, and liftoff processes are shown schematically in Figure 2.2a–e.

Alignment marks are made using a similar process as for electrodes. An extra isotropic etching[4] of the oxide layer (Figure 2.2f) is inserted after EBL (Figure 2.2a–c) to recess the marks so they do not interfere with the drawing method. The MAA/PMMA bilayer is not necessarily resistant to acid-based etches, so a simple test on uniform layers was completed to verify the suitability of this process for noncritical features. The results, listed in Table 2.3, show noticeable loss of MAA while the PMMA is largely unaffected. As before, the metal marks are defined by shape of the PMMA windows (Figure 2.2g–h) so the additional undercut only serves to widen the small trenches at the edges of the marks.

Table 2.3: Initial film thickness t_0 and etching rate in 10:1 buffered oxide etch for PMMA, MAA, and SiO_2

Material	t_0	Δt (60 s)
PMMA	1230 Å	−50 Å
MAA	3710 Å	−260 Å
SiO_2	2900 Å	−500 Å

[4] Immersion in 10:1 buffered oxide etch for 60 s

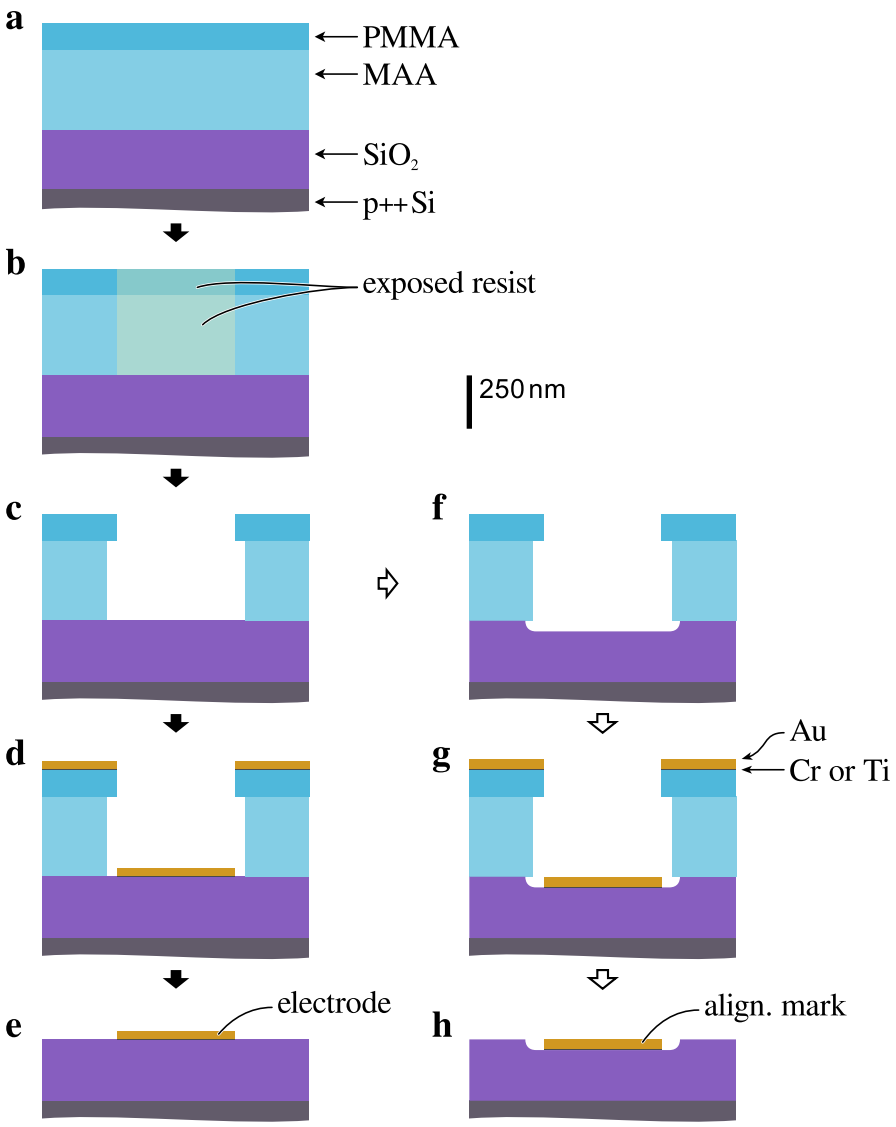


Figure 2.2: Electrode (a–e) and alignment mark (a–c, f–h) process flow. Vertical dimensions are to scale. (a) Substrate covered by bilayer MAA/PMMA resist. (b) Electron-beam exposed pattern. (c) Mask formed by development. (d) Cr/Au or Ti/Au metal film evaporated across entire substrate. (e) Electrode after liftoff. (f) Isotropically etched well. (g) Cr/Au or Ti/Au metal film evaporated across entire substrate. (h) Alignment mark flush with substrate surface

2.4.2 Alignment

In the context of EBL alignment refers to the process of transforming a lithography pattern such that overlays with features already on the substrate. The commonly used *Nanometer Pattern Generation System* (*NPGS*) [35] is capable of generating the necessary transformation matrix given four manually supplied reference points. (An example pattern and screen capture showing the process appear in appendix A.) While alignment normally goes smoothly, producing accuracy on the order of the

≈50 nm pixel size of the alignment windows, it is worth considering the process in more detail to be prepared in the event that problems do arise (and perhaps to have something to contemplate while the SEM is writing).

There are three coordinate systems—real space x-y, pattern space x_p-y_p, and writing space x_w-y_w—which may be rotated or skewed relative to one another.[5] Here the predefined alignment marks and graphene (obviously) exist in real space while the electrode design, made in a computer-aided design (CAD) program, exists in pattern space. The two are related by

$$\vec{r}_p = \mathbf{T_m}\vec{r}, \tag{2.1}$$

where $\vec{r}$ is the real position of a feature and $\vec{r}_p$ is the equivalent location in the CAD program. $\mathbf{T_m}$ differs slightly from the identity matrix depending on the calibration of the microscope used to measure the existing features. Given a raw pattern, the SEM will reproduce it at the surface of the sample subject to systematic distortion $\mathbf{T_s}$ defined by

$$\vec{r}_w = \mathbf{T_s}\vec{r}_p, \tag{2.2}$$

where $\vec{r}_w$ is a *written* feature's physical location.

In the simplest case, the goal of alignment is to make x_w-y_w overlay perfectly onto x-y. *NPGS* will calculate the optimal matrix $\mathbf{T_a}$ such that

$$\vec{r}_w = \mathbf{T_s}\mathbf{T_a}\mathbf{T_m}\vec{r} = \vec{r} \tag{2.3}$$

for the specified alignment points. The weakness of this approach may be demonstrated by considering a poor microscope with $\mathbf{T_m} \not\approx \mathbf{I}$.[6] A square shape in pattern space, while still guaranteed to match the preexisting features, will become nonsquare in writing space.

The process may be improved by splitting the alignment into two parts, effectively finding $\mathbf{T_m}$ and $\mathbf{T_s}$ independently. $\mathbf{T_m^{-1}}$ may then be applied to the pattern before loading it into *NPGS* such

[5] Trivial origin offsets are ignored in this exposition.

[6] The typical deviation of $\mathbf{T_m}$ from $\mathbf{I}$ depends on the type of microscope. One would expect optical microscopes to have very little distortion but some uncertainty in the overall magnification. An AFM may produce significant distortion due to drift in its piezoelectric X-Y scanner.

that $\mathbf{T_a} = \mathbf{T_s}^{-1}$. In practice the first alignment is performed by using image processing software to match alignment marks with their known locations. This assumes that the alignment grid was written accurately to begin with, a safe assumption when using a dedicated EBPG for that step.

2.5 Electron transport measurements

Electron transport measurements must be matched to the energy scale and time scale of the phenomena being probed. The thermal energy scale k_BT is 26 meV at 300 K and 121 μeV at 1.4 K, such that mV-scale DC biases are appropriate at room temperature while AC lock-in techniques are required a liquid helium temperatures. Switching measurements, on the other hand, require volt-scale biases with large bandwidth to capture ms- or μs-scale events. Three measurement setups common to the experiments in this work are described. Special attention to electrostatic discharge and grounding was paid throughout.

Current-voltage characteristics and gate voltage sweeps were recorded for many graphene devices at room temperature immediately after fabrication. These measurements were conducted in vacuum to minimize doping by adsorbed water vapor [36]. Vacuum conditions were also necessary for the electrical breakdown process described in chapter 3. The schematic for a typical setup is shown in Figure 2.3. A computer running custom software (*Mezurit 2*, described in detail in chapter 5) equipped with a National Instruments (NI) data acquisition (DAQ) card supplies a bias

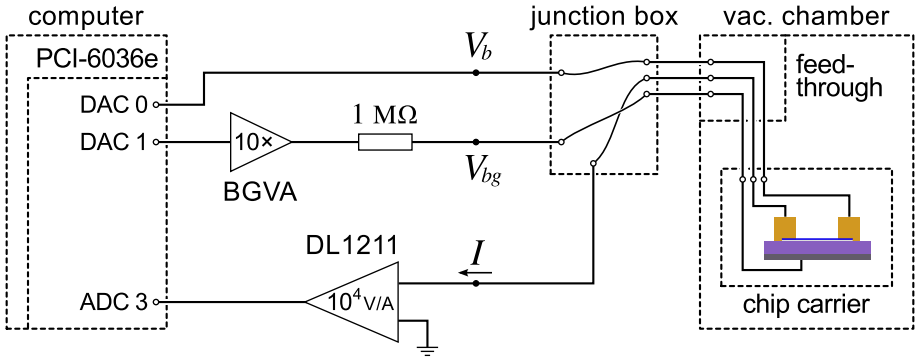


Figure 2.3: Circuit schematic for room temperature DC measurement, in vacuum[7]

[7] *Mezurit 2* implements *virtual channels*, making it easy to configure the DAQ card for any measurement. In this case, the three channels, X_0, X_1, and X_2, are as shown below. Note that SI prefixes are handled automatically such that current I will be displayed in μA without needing to modify X_2.

V_b	(V)	$X_0 = \mathrm{DAC}(0,0)$
V_{bg}	(V)	$X_1 = \mathrm{DAC}(0,1) \cdot 10$
I	(μA)	$X_2 = \mathrm{ADC}(0,3) \cdot 10^{-4}$

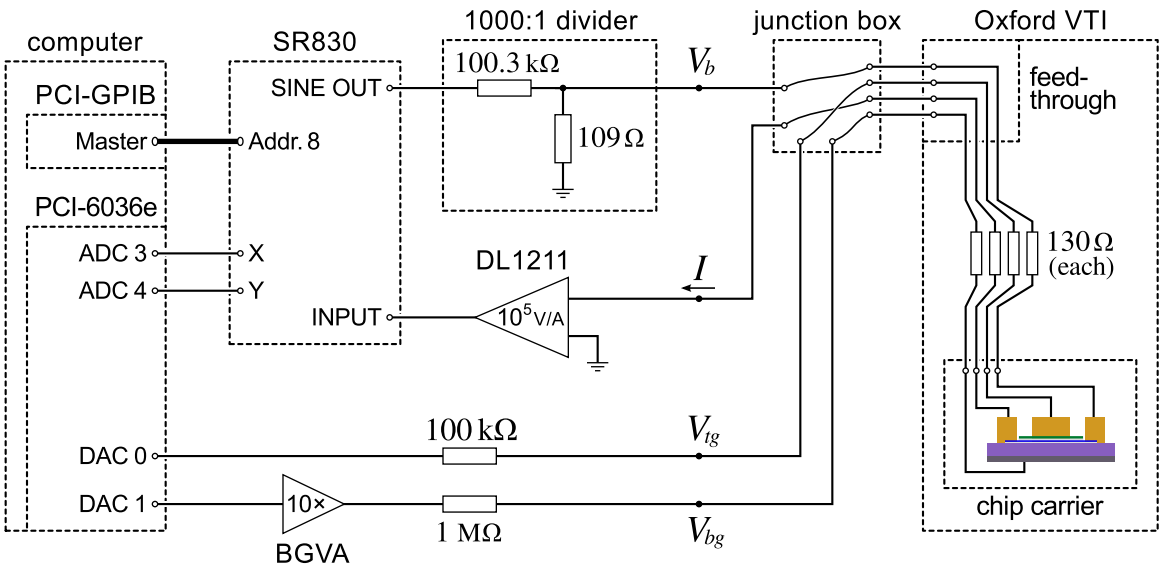


Figure 2.4: Circuit schematic for low-temperature AC measurement[8]

voltage V_b to a two-terminal graphene device placed within a sealed, evacuated chamber through a junction box and electrical feed-through. The back-gate voltage V_{bg} is supplied in proportion to a second analog output voltage by the custom bipolar gate voltage amplifier (BGVA) described in appendix B. Current I is measured by a DL Instruments current preamplifier which outputs a voltage to be read on one of the DAQ card's input ports.

Low-temperature measurements of linear response conductance were carried out in an Oxford Instruments variable temperature insert (VTI) using low-frequency AC lock-in techniques. The sample is such systems is cooled by evaporation of liquid helium flowing through a needle valve into the sample space, which is pumped down to 1 mbar. The circuit schematic is shown in Figure 2.4. In this case the V_b is necessarily supplied by the Stanford Research Systems SR830 lock-in amplifier and the measured current response is transmitted to the DAQ card as analog voltages proportional to its X and Y components. The DC top-gate and back-gate voltages V_{tg} and V_{bg} are supplied, as before, by the analog outputs of the DAQ card. The lock-in output level and input sensitivity are controlled by *Mezurit 2* over a general purpose interface bus (GPIB) connection.

[8] The virtual channels for this case are listed below. Note that GPIB functions SR830_SineOut and SR830_SineOut and SR830_SensIn are bidirectional such that they may be set in *Mezurit 2* or changed on the control panel of the lock-in.

V_b	(μV)	$X_0 = \mathrm{SR830\_SineOut}(0,8) \cdot 109/(109 + 100.3 \times 10^3)$
I_x	(nA)	$X_1 = \mathrm{ADC}(0,3)/10 \cdot \mathrm{SR830\_SensIn}(0,8) \cdot 10^{-5}$
I_y	(nA)	$X_2 = \mathrm{ADC}(0,4)/10 \cdot \mathrm{SR830\_SensIn}(0,8) \cdot 10^{-5}$
V_{tg}	(V)	$X_3 = \mathrm{DAC}(0,0)$
V_{bg}	(V)	$X_4 = \mathrm{DAC}(0,1) \cdot 10$

Chapter 3 describes bias-induced switching in graphene nanogaps on the order of 10^{-4} s or faster. Capturing the switching process therefore requires faster data acquisition than the 0.1–1 kHz used for low-speed DC measurements. *Mezurit 2* is capable of recording points up to 1 MHz in "scope-mode" but is limited to kHz-scale waveform generation. Thus, an external instrument such as the Stanford Research Systems DS345 function generator is needed to apply high-quality waveforms to the device. The DS345 was chosen for its arbitrary waveform capability, which allows the user to supply a digital description of the desired output as a function of time. Pulse trains were defined in simple text files and converted into the necessary format using *awcgen* (usage instructions in appendix C) before being programmed into the DS345 over RS-232.

The measurement setup is shown in Figure 2.5. The DS345 is configured to wait for a signal via the TTL-compatible[9] trigger input before running one cycle of the pulse train. On each run, *Mezurit 2* sends a 5 V trigger signal and simultaneously begins high-speed acquisition up to 100 kHz.[10] The bias voltage V_b is fed back to the DAQ card along with the measured current I to produce a synchronized dataset.

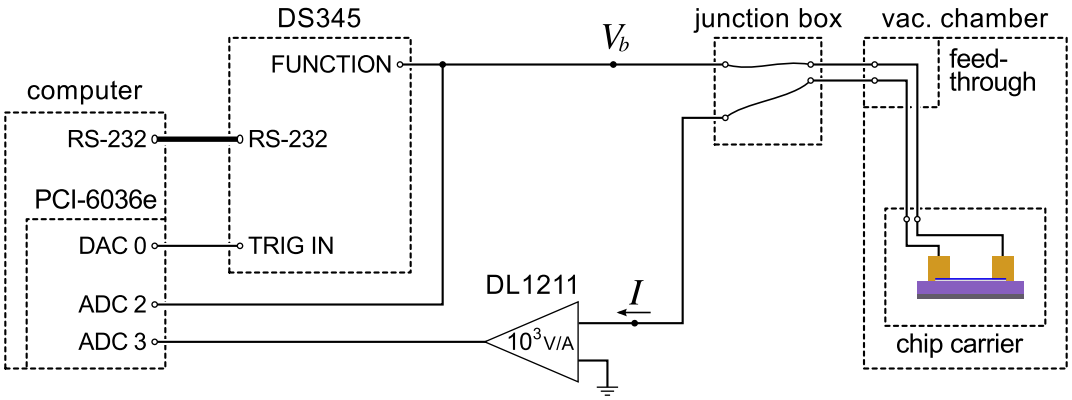


Figure 2.5: Circuit schematic for higher-speed measurement[11]

[9] TTL (transistor-transistor logic) allows voltage range of 0–5 V with 0–0.8 V defined as 0 and 2–5 V defined as 1.

[10] The NI PCI-6036e is capable of 200 kS/s, so for two channels the maximum rate is 100 kHz. The DL 1211 is also a factor —its 3 dB bandwidth is 60 kHz at 10^3 V/A and falls to 4 kHz at 10^9 V/A [37].

[11] The virtual channels for this case are listed below. Note that the trigger channel need not be saved to disk.

V_{trig}	(V)	$X_0 = \mathrm{DAC}(0,0)$
V_b	(V)	$X_1 = \mathrm{ADC}(0,2)$
I	(mA)	$X_2 = \mathrm{ADC}(0,3) \cdot 10^{-3}$

2.6 Future work

Improvements to existing fabrication and measurement techniques are an integral part of nanoelectronics research. There is a trade-off, however, between automating and parallelizing these techniques and focusing on the Science, so to speak. In practice the appropriate balance can be hard to find and may shift as new techniques become available or drop in time or monetary cost. Several possible future directions are highlighted:

- Aligned layer transfer [38], while itself an involved process, offers more control and possibly better yield.
- The expanded use of CVD graphene and parallel fabrication methods including photolithography would produce a much greater number of devices per unit time. This would enable more robust statistics and in general larger experiments. For example, a greater breadth of functionalization methods for graphene sensors could be tested at once.
- Transport measurements could be automated to a degree approaching that of industrial parametric test systems. A reasonable starting point would be a computer-controlled array of mechanical relays to replace the standard manual junction box and preamplifiers with remotely adjustable gain.[12]

[12] Auto-ranging digital instruments must be used carefully to avoid endangering delicate samples.

Chapter 3

Atomic-scale switching in graphene nanogaps

3.1 Introduction

The ultimate limit for the miniaturization of electronics is the atomic scale. Reaching this limit will require novel materials as well as new paradigms for device operation and architecture. Graphene has rapidly emerged as an exceedingly promising novel electronic material [1, 10, 39, 40] for scaling beyond-complementary metal oxide semiconductor (CMOS) circuitry and architecture. Compared to silicon, graphene has far superior mobility [1, 39, 41–44], thermal conductivity [45] and current-carrying capabilities [46], and may support room temperature ballistic transport [10]; yet like silicon, it has a planar geometry that enables multilayer device architecture and simplifies future integration with CMOS technology. However, the absence of a band gap poses a major challenge for graphene electronics, as this limits the on/off ratio of digital devices. Recent progress has been made towards fabrication of graphene nanoribbon field effect transistors with high on/off ratios [47–49]; nevertheless, this approach presents several additional challenges such as precision control of the ribbon width and the atomic structure of the graphene edges. Switching behavior with high on/off ratio may also be promoted via chemical modification of graphene, though improvement in cycling capability remains to be developed [50].

This chapter describes the fabrication and operation of graphene switches, and their application for nonvolatile information storage. Their behavior is interpreted using a model of electric field-driven motion of atomic chains of carbon. These switches are fabricated by creating nanoscale gaps using electrical breakdown of graphene sheets—a reliable self-limiting process that avoids the need

for advanced lithographic techniques, similar to that employed in metallic wires [51, 52]. Applying appropriate bias voltage pulses switches the gap conductance between high (ON) or low (OFF) conductance states. Remarkably, these switches are extremely robust; they have operated for many thousands of cycles without degradation, with the conductance state easily persisting for greater than 24 hours, and perhaps indefinitely. As an example of potential applications, a prototype circuit was built to demonstrate information storage based on the combination of two or more switches using rank coding, in which information is stored by the relative magnitudes of device conductance in a memory cell. Such non-volatile, robust, atomic switches based on graphene, which are suitable for integration with CMOS as well as monolithic integration with graphene electronics, are promising as components for atomic-scale devices.

3.2 Device fabrication

Two-terminal graphene devices are fabricated using standard techniques [22], as detailed in chapter 2. Briefly, graphene sheets are transferred from a high-quality graphite source[1] to a heavily p-doped silicon substrate with a 290 nm layer of thermally grown oxide. One- or two-layer graphene sheets are identified by color interference using optical microscopy, and then sorted by shape and size. Sheets which are roughly rectangular in shape and have a width of less than a few μm (to limit the required breakdown current) are made into devices by attaching metallic (Cr/Au or Ti/Au) leads using electron-beam lithography and thermal evaporation. No etching step is required given the prevalence of usable sheets produced by the graphene deposition process.[2] Figure 3.1a shows an electron micrograph of a completed sample containing five devices.

3.3 Measurements

The samples are placed within a vacuum chamber for room-temperature transport measurements, which is then evacuated to $\sim 10^{-7}$ Torr. High-vacuum conditions were found to be crucial to the breakdown process, presumably because this prevents atmospheric gases from reacting with the graphene.

[1] Highly-oriented pyrolytic graphite, NT-MDT, grade ZYA or Kish graphite, Covalent Materials

[2] This work requires only a few devices per sample. An etching process would be necessary to produce many devices from a single large graphene sheet, and is not expected to affect the behavior of the devices.

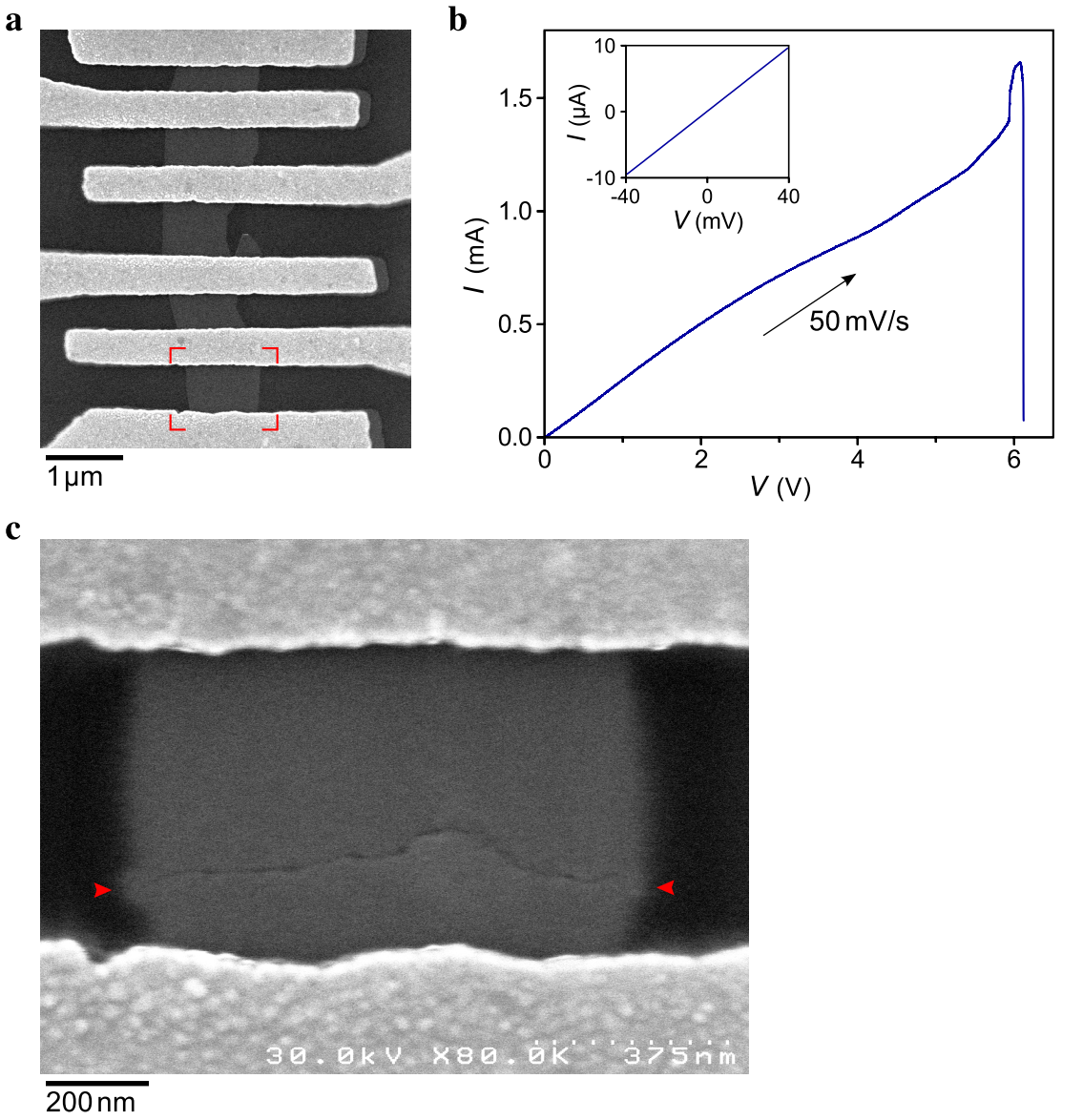


Figure 3.1: Device images and nanogap formation for a single representative graphene device. (a) Electron micrograph of the sample before as-fabricated. The red markers indicate the specific device under test. (b) Current versus voltage during the electrical breakdown process. Inset: Low-bias I-V characteristic prior to breakdown. (c) Electron micrograph of the area indicated by the red markers in (a), showing the device after breakdown. The red arrows indicate the ends of the nanoscale gap.

3.3.1 Electrical breakdown

For each device, a baseline current-voltage (I-V) characteristic is first obtained up to a bias of 0.1 V. Next, electrical breakdown is performed by gradually increasing the bias voltage while monitoring the current. A standard DC measurement setup (detailed in chapter 2) is used for both measurements; bias voltage is supplied by a computer-based data acquisition card which also records the

current as measured by a DL Instruments 1211 current preamplifier.[3] Figure 3.1 demonstrates the process for a single, representative device (identified by the red markers in part a). The baseline I-V (Figure 3.1b, inset) demonstrates a before-breakdown conductance of $\sim$250 µS. The breakdown I-V is shown in Figure 3.1b. Starting from 0 V, the voltage is ramped up at a uniform rate of 50 mV/s. At a critical sheet current density of approximately 1.6 mA/µm, the measured current drops precipitously, indicating breakdown of the graphene sheet. Interestingly, the conductance I/V increases before breakdown, possibly due to current-driven annealing [53]. Figure 3.1c shows an electron micrograph of the device after breakdown, containing an distinct break across the graphene channel. Atomic force microscopy (AFM) confirms that a physical gap forms in the graphene. In the most narrow segments the gap's width is typically less than 10 nm, although the precise location of the closest approach is difficult to determine due to the finite resolution of AFM. Thus, the device after breakdown includes a nanoscale junction between two physically separated sheets of graphene.

3.3.2 Junction conductance

Following the breakdown step, the low-bias device conductance is much lower, ranging from $\sim$0.1 nS to $\sim$20 µS. Remarkably, this conductance can be reproducibly increased or decreased using additional voltage pulses. Figure 3.2 illustrates a single cycle of such conductance modification applied to the same device. These data were obtained using a function generator to supply the voltage pulses combined with "scope-mode" data acquisition as described in chapter 2. The upper trace displays the applied source-drain bias V, which is quickly ramped up as a function of time, and the lower trace displays the current response. When $V < 2$ V, the current is very low; when it reaches $\sim$2.5–4 V, the current abruptly increases, reaching a maximum of 0.65 mA up to $\sim$5 V. At still larger voltages, the device returns to its initial low-conductance state. The conductance exhibits little or no gate voltage dependence in either state. Thus, by applying voltages of different magnitudes, the graphene devices can be switched between relatively high-conductance, defined as "ON", and low-conductance, defined as "OFF", states, thereby functioning as voltage-programmable bistable switches or memory elements.

[3] This technique has a bandwidth of approximately 1 kHz.

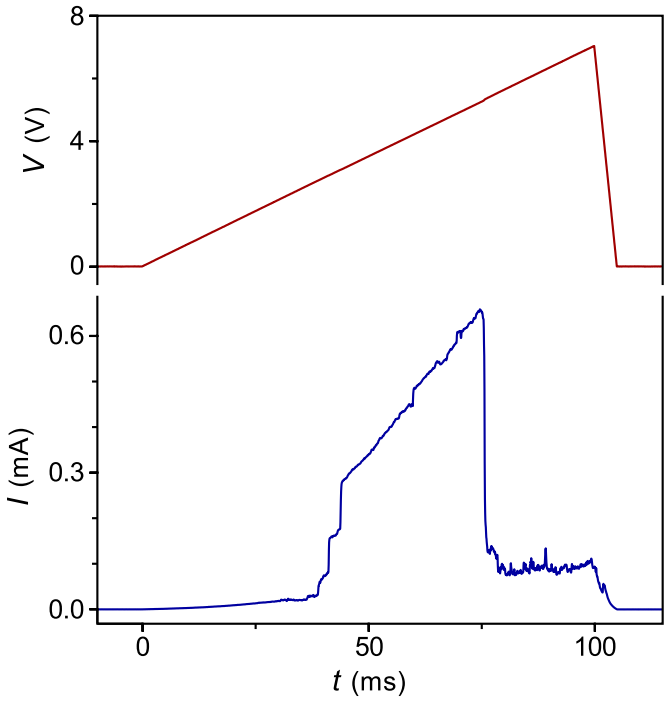


Figure 3.2: Switching behavior in a representative device. Applied junction voltage (upper trace) and junction current (lower trace) versus time for the device shown in Figure 3.1 after electrical breakdown, showing conductance recovery and subsequent reduction

Similar conductance switching has been observed in a number of systems, such as molecules [54], titanium oxides [55], and Ag/Ag_2S nanowires [56]. What sets these graphene switches apart are their unique combination of reproducibility, nonvolatility, and integrability with graphene electronics. The conductance state has been cycled up to 10^5 times without degradation. Once the devices switch, they remain in either ON or OFF states for greater than 24 hours at room temperature, without external maintenance voltages. Although a systematic study of continuous monitoring for longer time periods has yet to be performed, device conductance values have remained similar to their last written states after several weeks.

Figure 3.3 demonstrates repeated operation of the devices. The upper trace shows the programming and readout voltage versus time. A ramped pulse that reaches 4 V corresponds to a turn-on pulse that increases the device conductance, while a pulse with a maximum of 6 V corresponds to a turn-off pulse that decreases the device conductance. In between the ON and OFF pulses, low-bias readout pulses are applied. The lower trace shows the current response to the applied voltage versus time. Note that a large difference in current is observed during the readout pulses of the ON and OFF

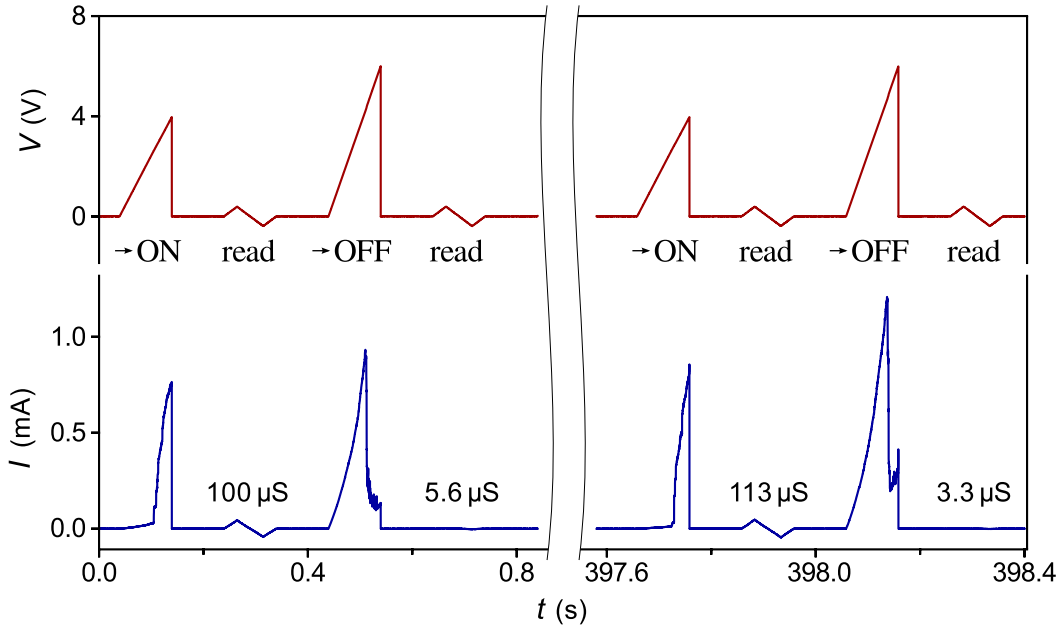


Figure 3.3: Repeatable programming over hundreds of cycles. Upper trace: Voltage applied to the junction versus time. A ramp with a peak value of ∼4 V corresponds to a turn-on pulse, while a ramp with a peak value of ∼6 V corresponds to a turn-off pulse. A small, sawtooth-shaped readout pulse is applied after each write to determine the low-bias junction conductance. Lower trace: Current flow through the junction, with the extracted low-bias conductance labeled above each readout pulse

states, although similar current levels flow during the writing process. The ON state conductance is typically ∼100 µS, while the OFF state conductance is typically ∼5 µS. The right-hand section of the plot shows data from the end of the measurement, which comprised ∼500 switching cycles. The device behavior is virtually identical to that shown on the first cycle, demonstrating exceptional reproducibility.

The observed switching behavior of graphene devices that have been electrically broken-down is quite surprising, especially given the robustness and reproducibility of this phenomenon. One possibility is that the conductance occurs along a small graphene ribbon that bridges the contacts. However, small ribbons are expected to have a band gap [47–49], with a strong gate voltage dependence in conductance, which is not observed. The absence of a gate voltage effect also excludes switching mechanisms arising from the storage of charge in traps in the oxide layer, or from formation of a small metallic island that may function as a quantum dot.

3.3.3 Time-resolved switching

To gain further insight into the switching mechanism, the time-resolved transition from the OFF to ON state was captured, again using a function generator and "scope-mode" data acquisition. The voltage V was quickly increased while monitoring the current, as in Figure 3.2. In this case, the conductance I/V is plotted versus V for the voltage range where switching normally occurs (Figure 3.4a). The plot shows well-defined steps in I/V, with magnitude $\sim G_0$. These characteristic step sizes, which include a series resistance from the graphene sheets, can vary by factor of order unity from device to device. Here $G_0 = 2e^2/h$, the conductance quantum, where e is the electron charge, and h is Planck's constant. Since G_0 is the conductance of a spin-degenerate one dimensional conductor, e.g., a linear chain of gold atoms [13], observation of these steps suggests that the device conductance states are likely multiples of highly transmitting quantum channels.

These data were recorded using a 10 kHz sampling rate to attempt to capture the switching process. Although for technical reasons[4] it is difficult to visualize the time scale of the switching in Figure 3.4a, careful inspection of the raw data (not shown) reveals that the conductance value transitions between the step plateaus in no longer than 2–3 $\times$ 10^{-4} s. The actual transition is probably

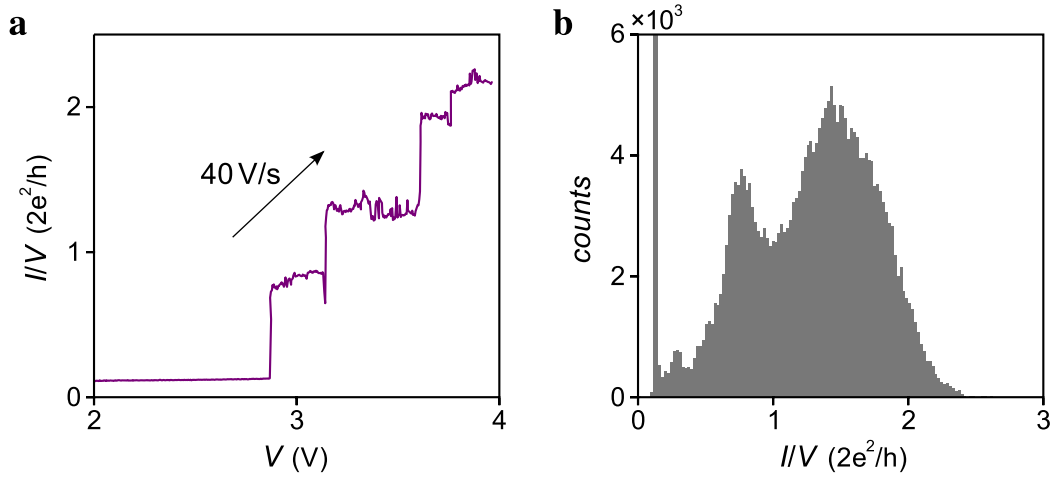


Figure 3.4: Conductance steps and histogram of conductance during switching. (a) Conductance I/V versus bias V recorded while the device switches from the OFF to ON state. A sequence of steps of magnitude $\sim 2e^2/h$ are observed. (b) Histogram of conductance values showing two major peaks at approximately integer multiples of $2e^2/h$

[4] Both the voltage and current are recorded as a function of time. When a transition occurs, the increased current causes a small decrease in the voltage measured across the device due to series output resistance of the function generator. Thus, the voltage scale of the plot is not perfectly linearly related to the time scale.

faster still; the time resolution of this measurement is limited by the bandwidth of the hardware.

To further investigate this phenomenon, a histogram of the conductance was assembled from 1000 cycles, shown in Figure 3.4b. The histogram exhibits two main peaks, separated by $\sim 2e^2/h$. These well-defined peaks, as well as the conductance steps, strongly resemble those found in the breaking process of mechanically controlled break junctions [13, 57], in which a linear chain of atoms forms between two closely spaced metallic electrodes. These results strongly suggest that the conducting pathway bridging the graphene edges is atomic in scale.

3.3.4 Switching rate

Additional information about the switching mechanism is provided by the voltage dependence of the switching rate. A square pulse with a given voltage magnitude is applied to the device in the OFF state, and waiting time t, defined as the time elapsed before the first sharp increase in conductance, is recorded. Because of the stochastic nature of the switching, t follows a statistical distribution. Data from approximately 800 switching cycles were compiled into histograms. The upper panel in Figure 3.5a shows a histogram of the measured t at V = 2.8 V. The peak at t ~200 ms indicates the most probable time scale at which switching occurs. The lower panel shows similar data, but with a larger bias V = 3.4 V. The peak in probability distribution visibly shifts towards zero, indicating the high probability of switching at relatively short time scales. Moreover, these observed switching rates are found to be strongly temperature dependent, increasing by a factor of ~3 for every 10 K increase in temperature.

The strong temperature and voltage dependence indicates a thermally activated process. In the simplest picture, such a process can be described by a single rate

$$\Gamma = \nu e^{-\Delta(V)/k_B T}, \tag{3.1}$$

where $\Delta(V)$ is the bias-dependent energy barrier, k_B is Boltzmann's constant, T is the temperature, and ν is an attempt frequency. At fixed temperature and bias, the rate is a constant, hence the waiting time would follow Poisson statistics, yielding a waiting probability per unit time $P(t) = \Gamma \exp[-\Gamma t]$. While this could account for the behavior shown in the lower panel, it cannot account for the maximum observed at finite waiting time in the upper panel.

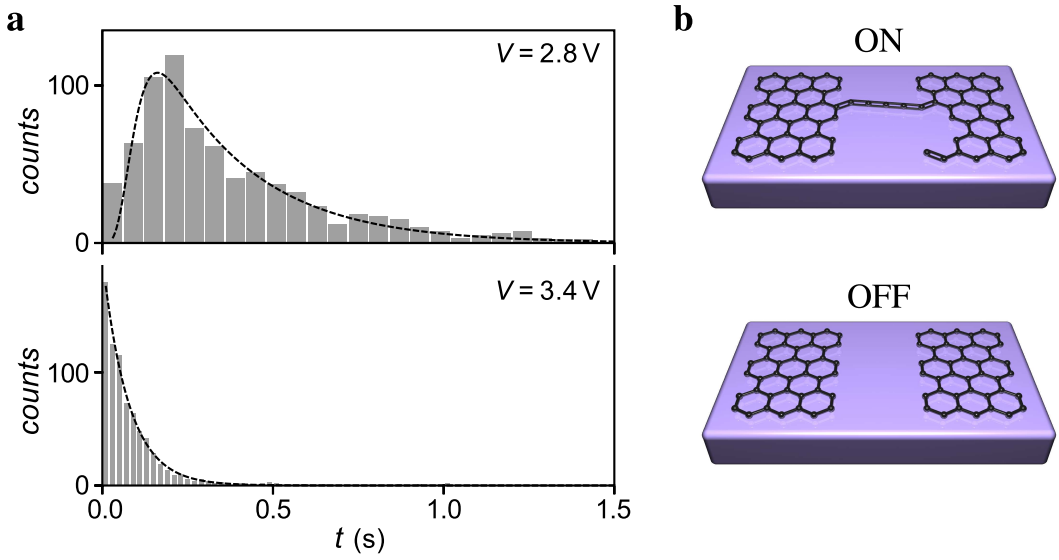


Figure 3.5: Statistical analysis of the switching dynamics and schematic of nanogap ON and OFF states. (a) Histograms of waiting times for the first conductance-increase step for two different bias voltages. (b) Proposed schematic atomic configurations in the ON and OFF states

3.4 Carbon chain model

Given the results above, the conductance switching is proposed to occur by formation of linear chains of carbon atoms that bridge the gap under a strong electric field (Figure 3.5b). The formation of such chains was reported earlier to account for the strong field emission current from the open end of multiwalled nanotubes [15], and the associated unraveling process was studied theoretically [58]. Theoretical calculations predict that these chains are metallic without significant Pierls distortion, and have the cumulene structure where each carbon atom in the chain is double-bonded to its neighbor [59]. Their expected conductance is $\sim 2e^2/h$ [17]. If multiple chains bridge the gap as the turn-on pulses are applied, this would be expected to produce $\sim 2e^2/h$ steps in the conductance, as observed. Moreover, calculations show [17, 60] that the conductance of these wires is expected to be approximately independent of the Fermi level, therefore accounting for the lack of an observed gate voltage effect. Finally, the voltage dependence of the distribution in t (Figure 3.5a) can be readily explained by considering a two-step process—the first step consists of n unlinking events, each of which corresponds to a carbon atom being "unzipped" from the graphene sheet, and has a

rate Γ_1; the second step is the binding of the chain which extends all the way across the gap, with rate Γ_2. Both these rates should have a similar form to equation 3.1. The distribution $P(t)$ for the total waiting time would then be given by the convolution of the waiting time for an n-step Poisson process with rate Γ_1, known as the Erlang distribution, and a Poisson process with rate Γ_2. Thus,

$$P(t) = \frac{\Gamma_1^n \Gamma_2}{(\Gamma_1 - \Gamma_2)^n} e^{-\Gamma_2 t} - \sum_{m=1}^{n} \frac{\Gamma_1^n \Gamma_2}{(m-1)!} t^{m-1} e^{-\Gamma_1 t} \frac{1}{(\Gamma_1 - \Gamma_2)^{n-m+1}}. \tag{3.2}$$

Fitting equation 3.2 to the data shown in Figure 3.5a gives $\Gamma_1 = 51\ \mathrm{s}^{-1}$, $\Gamma_2 = 3.6\ \mathrm{s}^{-1}$ for the upper panel, and $\Gamma_1 = 4300\ \mathrm{s}^{-1}$, $\Gamma_2 = 13\ \mathrm{s}^{-1}$ for the lower panel. A value of $n = 5$ was assumed based on the nm-scale gaps observed, which corresponds to a $\sim$1 nm long carbon chain. The quality of the fit was not sensitive to the precise value of n, which indicates considerable uncertainty in the value of Γ_1. Nevertheless, the time constants observed are consistent with eV-scale energy barriers.

This model of extending carbon chains coherently and comprehensively accounts for all the experimental observations: the lack of gate dependence on conductance, the $2e^2/h$ conductance steps, as well as the voltage and temperature dependence of waiting time distribution. Taken together, the results strongly suggest that the devices switch by the formation and breaking of chains of carbon atoms; the conductance state of the device therefore changes by the motion of atoms rather than by electric charge. In addition, since the initial publication of these results [61] other workers[5] have demonstrated similar switching behavior in nanogaps made from *suspended* graphene devices [62]. Although their results differ in some ways from those presented here, they do indicate that a nearby substrate is not an essential feature of a graphene-based switch.

3.5 Application to logic circuits

The properties of these junctions can be exploited to make logic gates and store information. However, one potential obstacle is the relatively modest on/off ratio (<100), while 10^5–10^6 is typically achieved in CMOS devices. The distributions of conductance for the OFF and ON states was measured by applying 4000 8 V square OFF pulses and 4000 4 V square ON pulses, each 100 ms long,

[5] This work was carried out by H. Zhang et al. at the University of California, Riverside with collaboration from B. Standley.

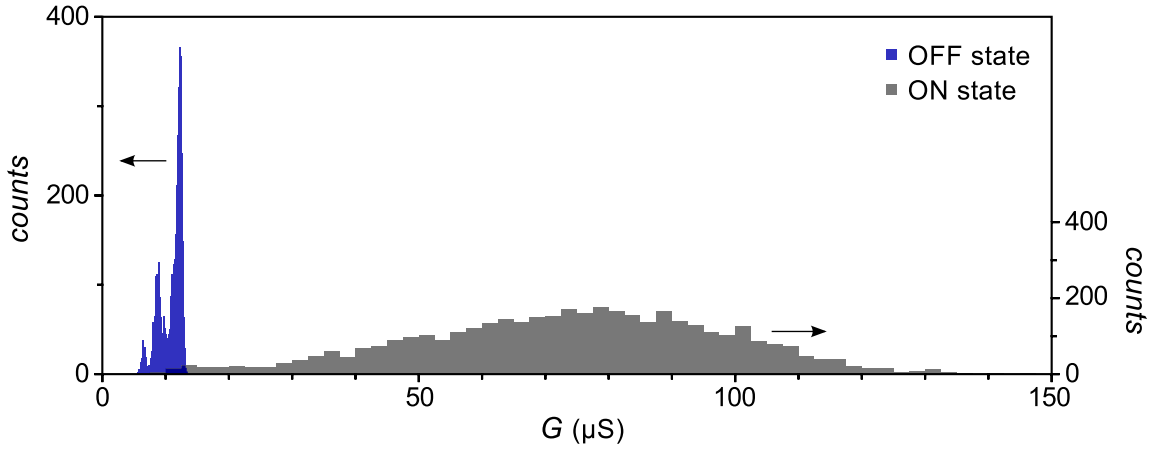


Figure 3.6: Low-bias junction conductance measured immediately after ON and OFF pulses. The left-hand axis applies to the OFF state distribution and the right-hand axis applies to the ON state distribution. The relative scales are adjusted for clarity given the different bin sizes of the two histograms (area as probability density is not directly comparable).

to a device in an alternating fashion. The low-bias conductance was measured after each writing pulse, and the results were compiled into histograms, shown in Figure 3.6. The maximum values of the two distributions lie 66 μS apart, but there is no significant separation between the tails. Thus, a graphene switch as a single memory element would have little noise margin. In addition, there is a small overlap in the 15 μS region which corresponds to a 0.5 % error rate.

3.5.1 Rank coding

One alternative approach is to utilize a novel memory cell based on rank coding [63]. A bit is stored not by the absolute value of the device conductance but by the comparison of the conductance of two or more devices in a cell. The information capacity for an N device cell is therefore $\log_2(N!)$ bits, which, for large N, exceeds even that of a conventional memory cell.

As a first demonstration of this concept, a rank-coded cell (RCC) was built using $N = 2$ graphene devices to store one bit. The two graphene devices, G1 and G2, are each connected in series to a 200 kΩ bias resistor to created two branches. A single readout voltage source V_R is connected to both branches in parallel, and each branch includes an individual write line and readout line connected between the device and bias resistor. During the writing phase, V_R is held at 0 V while

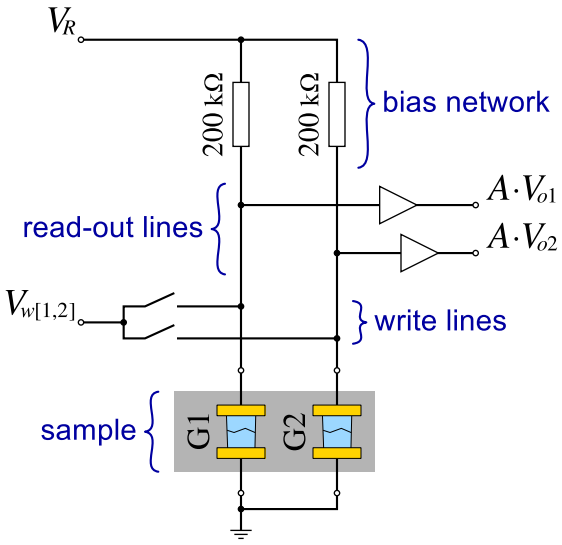


Figure 3.7: Circuit diagram of rank-coded cell

one of the write lines is connected through a multiplexer[6] to a voltage source, labeled V_{w1} or V_{w2} depending on the selected device. During the readout phase, write lines are left floating[6] while voltages V_{o1} and V_{o2} on the readout lines are recorded. In this way, the conductance states of the two devices can be inferred from their respective readout voltages $V_{o[1,2]} = V_R R_{[1,2]}/(R_{[1,2]} + 200\,\mathrm{k\Omega})$, where R_1 is the resistance of G1 and R_2 is the resistance of G2, without the need for complex current-measuring circuitry. A schematic of the device circuit is shown in Figure 3.7.

The logical value X of the cell is defined by the relative conductance of the two devices:

$$X = \begin{cases} 0 & \text{if } V_{o1} < V_{o2} \\ 1 & \text{if } V_{o1} > V_{o2} \\ \text{undef.} & \text{else} \end{cases} \tag{3.3}$$

The readout voltages may be sent to a comparator to encode X as a digital logic-compatible voltage, or, as in this proof-of-concept demonstration, plotted and compared visually.

Figure 3.8a demonstrates the operation of the memory cell. The top three traces plot input voltages V_{w1}, V_{w2}, and V_R versus time, while the bottom panel shows the readout voltages for both devices. The RCC demonstration begins with G2 in a more conductive state than G1 such that V_{o2} is "pulled" lower than V_{o1}, corresponding to $X = 1$. Voltage V_R is then decreased to zero, after which

[6] For this preliminary demonstration, the multiplexers and external switches were simulated using a junction box containing mechanical switches.

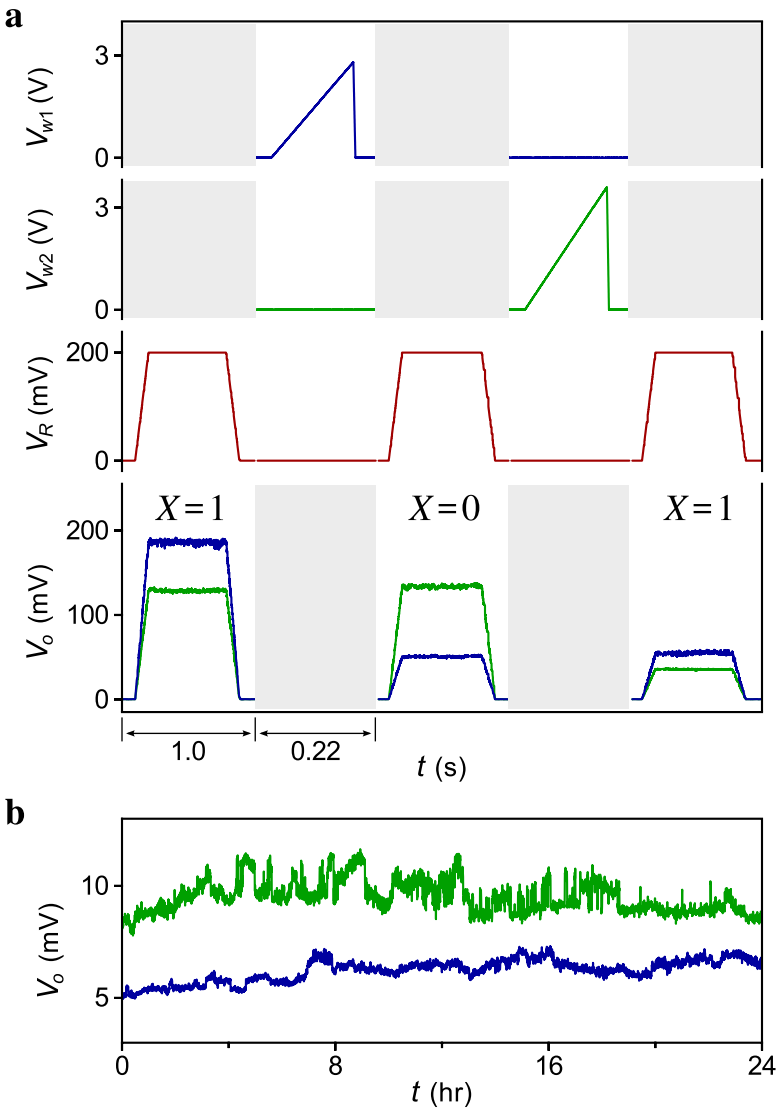


Figure 3.8: Demonstration of operation and retention time for the rank-coded cell described in Figure 3.7. (a) Write and read-out voltages versus time. Top two traces: Voltages applied to the write lines of device G1 and G2. Third trace: Readout excitation voltage. Bottom traces: Readout voltage for devices G1 (blue) and G2 (green). The logical state X is indicated above each readout. (b) Readout voltage versus time over a 24 hour period with no write voltages applied

a turn-on pulse is applied to G1, increasing its conductance such that $V_{o1} < V_{o2}$, flipping value of the memory bit. After another readout cycle to confirm the state of the RCC, a turn-on pulse is then applied to G2, which increases its conductance beyond that of G1,[7] switching X back to 1. Thus, the logical state of the RCC has been switched from 1 to 0 and back to 1 again.

[7] A slightly larger turn-on pulse is applied to G2 to promote increased conductance, ensuring that its conductance "leap-frogs" that of G1. In practice, this means the switches are not guaranteed to follow this pattern, so additional write pulses may be required.

In principle, the sequence above could be continued indefinitely by applying turn-off write pulses to reset G1 and G2 to their original low-conductance states. This could be completed one device at a time such that X either retains its value or not, depending on the desired outcome. Given the stochastic nature of the switching in graphene nanogaps, a more-robust RCC may require a controller circuit that reads X after each write to confirm that the turn-on or turn-off pulse had the intended effect. The stability of the stored information is demonstrated in Figure 3.8b, which shows the voltage on the two outputs read continuously for 24 hours, without voltages applied on the write lines. The voltage rank is maintained during the observation time, indicating the stability of the stored bit for extended time periods.

3.5.2 Switching energy

One important metric of a digital memory technology is the switching energy Q_{sw} required to flip one bit. This value can be roughly estimated by assuming a square writing pulse of height V_{sw} and width γt_{sw}, where t_{sw} is the characteristic time before switching and γ is a safety factor. The safety factor is necessary to accommodate the tail of the distribution of t_{sw} given the stochastic nature of the process, as illustrated for the OFF to ON transition in section 3.3.4. Given known preswitching current I_1 and postswitching current I_2, the switching energy is

$$Q_{sw} = V_{sw} t_{sw} \left[I_1 + (\gamma - 1) I_2 \right]. \tag{3.4}$$

Using typical values observed in the data used to generate Figure 3.6 and γ = 5 yields 70 µJ (3 µJ for γ = 1) when switching from OFF to ON[8] and 6 µJ (3 µJ for γ = 1) when switching from ON to OFF.[9] These values are significantly higher than for other recent architectures, for example 0.7 pJ in TiO_2 resistive memory [64], but may be greatly reduced in future work. Possible avenues of exploration include high-frequency ON pulses, higher-voltage OFF pulses, and intelligent controller circuits which adapt the writing voltage pulse to the device.

[8] OFF→ON: V_{sw} = 4 V, t_{sw} = 10 ms, I_1 = 80 µA, I_2 = 400 µA.
[9] ON→OFF: V_{sw} = 6 V, t_{sw} = 1.0 ms, I_1 = 550 µA, I_2 = 100 µA.

3.6 Future work

These results open the door for a number of directions of future research. Apart from forming the basis for nonvolatile, robust and CMOS-compatible memory devices, reconfigurable circuits and logic gates, these devices may enable detailed transport studies of individual carbon atomic chains. More generally, a simple, bottom-up method has been demonstrated to realize a novel nanoscale system: the graphene nanogap, which may be useful for a variety of scientific studies such as making single molecule transistors [65] or combined transport and scanned probe experiments. Although these graphene switches' writing speed is presently relatively slow (up to 100 ms), a dramatic increase in switching speed was found at elevated temperatures, indicating substantial potential for improvement. Furthermore, the devices could in principle be imprinted in dense arrays that are suitable for readout by scanned probe arrays [66], yielding a basis for large-capacity long-term information storage.

Chapter 4

Graphite oxide gate dielectric for field-effect transistors

4.1 Introduction

Graphene's excellent electronic, thermal, and mechanical properties make it an exciting novel material for many applications. Finding a compatible dielectric material remains a challenge due to graphene's sensitivity to environmental conditions, including doping and charge disorder. While significant progress has been made in adapting bulk insulators including Al_2O_3 [67, 68], HfO_2 [69–71], and lead zirconate titanate (PZT) [72], the use of layered materials is just beginning to be explored, for example, in carbon nanotube-graphite oxide-metal transistors [73] and graphene-boron nitride-metal transistors [74, 75]. Graphite oxide (GO) is of particular interest because it is compatible with solution-based processing and deposition [31, 76]. In addition, compared to boron nitride, GO has the added benefit of being graphene's native oxide, analogous to to silicon's native oxide, SiO_2. Like SiO_2, GO can be made from its parent material via oxidation [31], or by local oxidation [77], and is an electrical insulator. This correspondence suggests the possibility of a novel all-graphene-based field-effect transistor built with a graphene channel, graphite oxide gate dielectric, and graphene top-gate.

This chapter describes the fabrication and operation of capacitors and field-effect transistors (FETs) built using one-to-few layer graphene channels and 4–35 nm stacks of graphite oxide sheets as the gate dielectric layer. The carrier density in the graphene channel can be tuned by both a doped-Si back-gate and a local metal top-gate. Using a low-temperature fabrication process (with temperature T maintained below 120 °C), these devices show minimal gate leakage at room temperature

and breakdown electric field comparable to SiO_2, typically $\sim$1–3 $\times$ 10^8 V/m. Comparing the gate efficiency of the back and top-gates gives an estimated relative dielectric constant approximately equal to SiO_2, $\kappa \approx 4.3$. The successful operation of these devices demonstrates graphite oxide's utility as a layered insulator for graphene circuits which could be used in a multitude of configurations. This shows the promise of GO for use in future all-carbon circuits, which could be used for flexible electronics.

4.2 Device fabrication

Graphene-graphite oxide field-effect transistors (G-GO FETs) are fabricated by random deposition, as illustrated in Figure 4.1, using the techniques described in chapter 2. First, graphene sheets are deposited by mechanical exfoliation [22] onto a heavily p-doped silicon substrate capped with a 290 nm layer of thermally grown oxide. Next, graphite oxide flakes are made by oxidizing graphite powder,[1] following the Hummers method [32] as described in section 2.3. The graphite oxide flakes are then mixed with deionized water to form a suspension with concentration $\sim$1 mg/mL, which is stirred for at least one day to exfoliate the flakes into thin sheets. Stirring produces larger, thicker sheets than ultrasonication. The graphite oxide sheets are then deposited randomly onto the substrates [31, 33, 78] using a "drop-and-dry" method.

Overlapping regions of one-to-few-layer graphene and one-to-multilayer graphite oxide sheets are identified using optical microscopy. An example region is shown in Figure 4.2a. Note the purple color of the underlying graphene, which can be seen through the green-tinted graphite oxide.

4.2.1 Electrostatic force microscopy

The geometry can be confirmed using electrostatic force microscopy (EFM) [79–81], which gives a qualitative measure of the local conductivity at the surface of a sample [82, 83]. EFM is performed by scanning a mechanically oscillating probe tip over the sample at constant (time) average height[2] while a constant voltage bias is applied between the tip and sample. When the tip is located over a conductive region of the sample, the charge on the tip causes an image charge to form at the surface

[1] Sigma-Aldrich graphite powder, <20 µm, synthetic

[2] Knowledge of the local topography is needed to maintain a constant height. This information is determined by conventional atomic force microscopy (AFM), immediately before each "line" of the EFM image is recorded. (The voltage bias is turned off during AFM to avoid damaging the sample.)

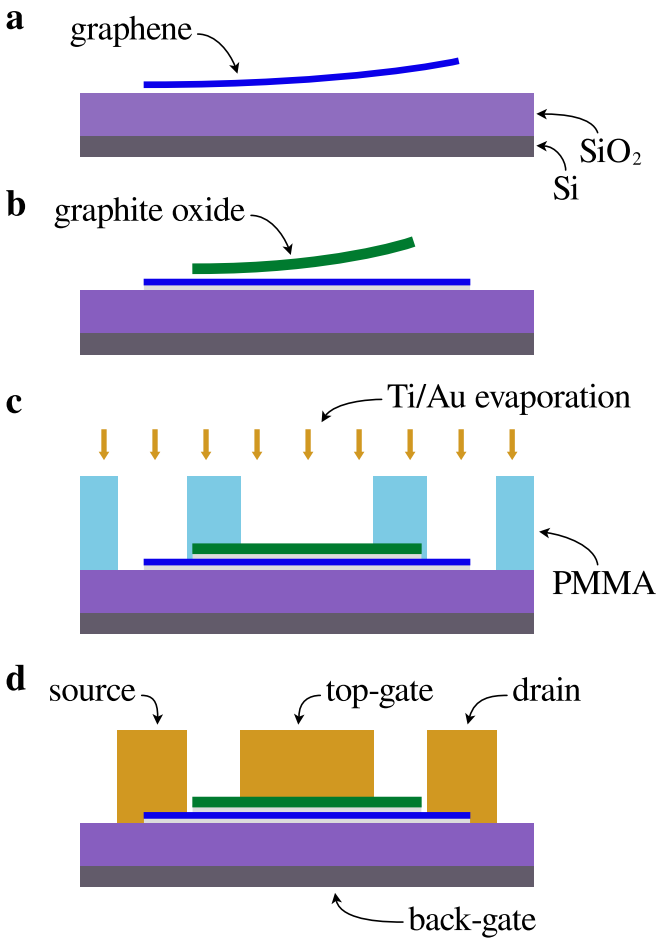


Figure 4.1: Graphene-graphite oxide process flow (not to scale). (a) Graphene is deposited onto an oxidized Si wafer (violet). (b) Graphite oxide is then deposited from an aqueous solution. (c) Metal source, drain, and top-gate contacts are then deposited using electron beam lithography and electron beam evaporation. In practice, bilayer resist (not shown for clarity) is used to promote metal liftoff. (d) Schematic diagram of completed device

of the sample. This creates an electrostatic force on the cantilever, producing a phase shift in the mechanical oscillation. The phase shift versus tip position for the region shown in Figure 4.2a is shown in Figure 4.2b. The region covered by graphite oxide, outlined by a white dashed line, shows little or no phase shift compared with the substrate except for the parts covering graphene, which are darker. This indicates that as-deposited graphite oxide is much less conductive than graphene. Separate transport measurements on as-prepared GO flakes typically showed little or no measurable lateral conductance. One such device is shown in Figure 4.3. No signal was measured by a current amplifier configured for nA sensitivity for a voltage bias of 0.5 V, indicating that the resistance is at least 10–100 GΩ.

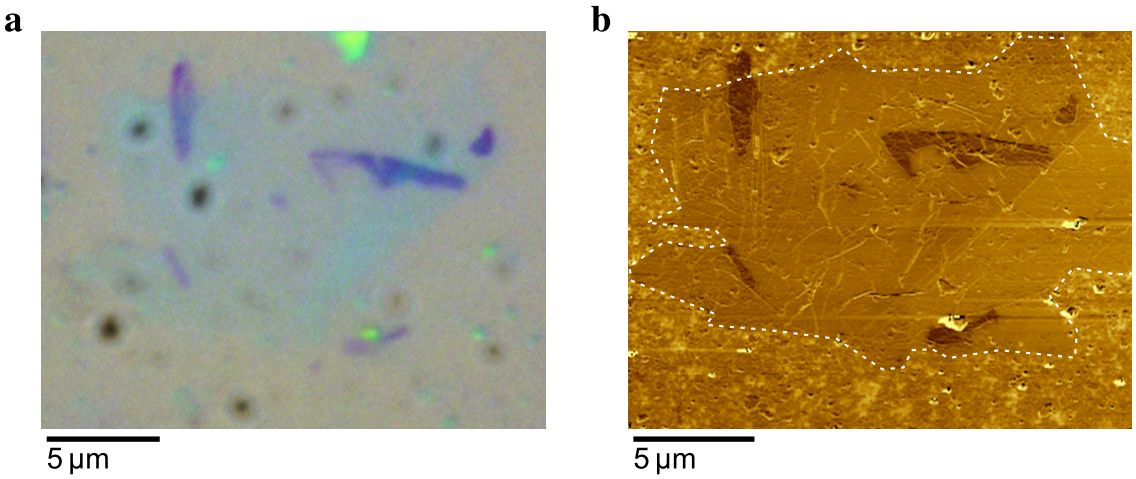


Figure 4.2: Images of graphene covered by graphite oxide. (a) Optical micrograph under white light of several graphene sheets partially covered by graphite oxide. Graphene appears violet, graphite oxide light blue-green. (b) EFM phase image of the site shown in (a), with the graphite oxide region outlined in white

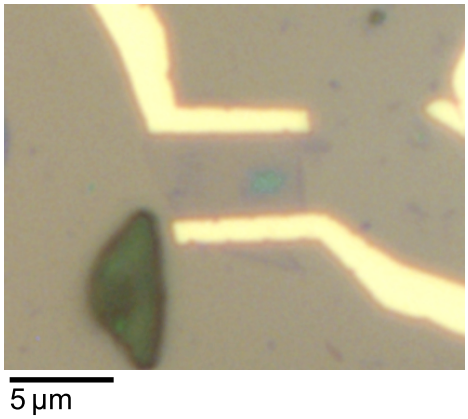


Figure 4.3: Optical micrograph of graphite oxide flake contacted by metal electrodes

Overlapping G-GO regions must be suitable for making a FET, typically with a uniform µm-scale graphene strip covered by a continuous graphite oxide sheet but with uncovered ends for the source and drain contacts. In practice, such sites can take up to several hours to locate. To complete a device, metallic (Ti/Au) source, drain, and top-gate electrodes are created using electron-beam lithography and electron-beam evaporation. A finished device with capacitor geometry is shown in the electron micrograph in Figure 4.4a and an atomic force micrograph of a FET is shown in Figure 4.4b.

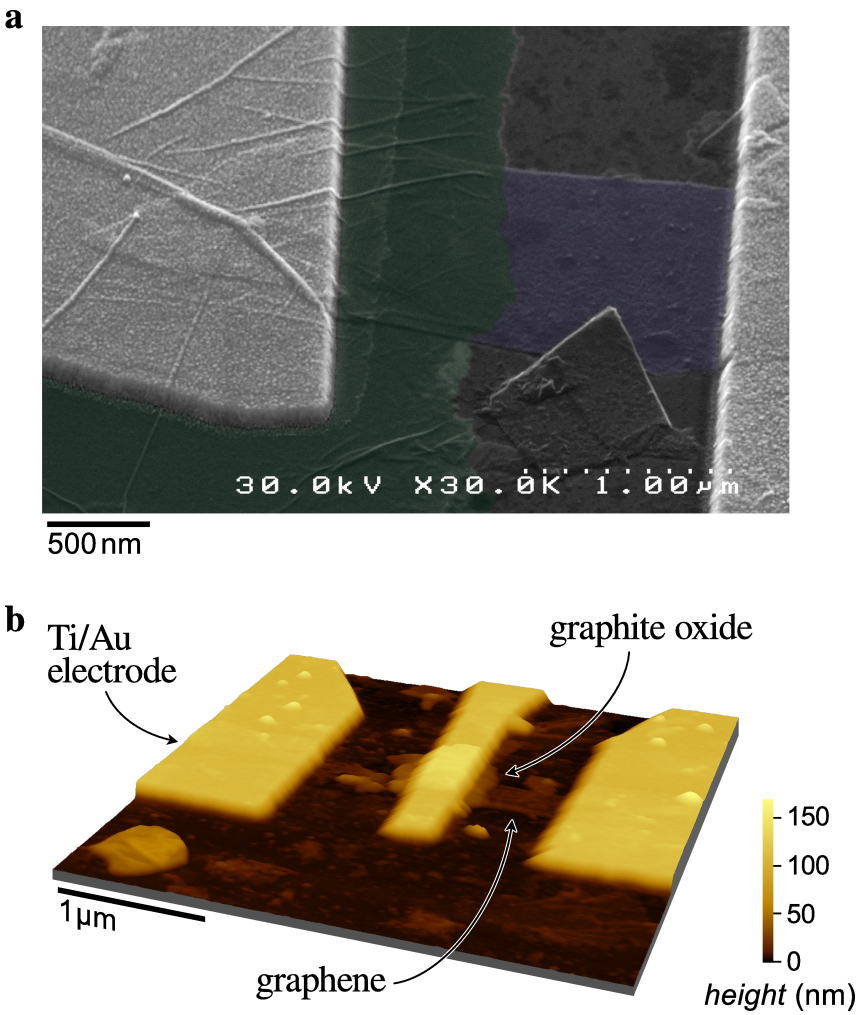


Figure 4.4: Images of G-GO devices. (a) False-color electron micrograph of G-GO capacitor. The lower graphene electrode (shaded blue) is contacted on the right by a metal electrode. The graphite oxide layer (shaded green) is sandwiched between the graphene and the metal electrode at left. (b) False-color atomic force micrograph of a G-GO FET

4.2.2 Low-temperature processing

Thermal reduction of graphite oxide results in at least partial recovery of graphite's electrical conductivity [33]. Lateral transport measurements on GO devices undergoing heat treatment (420 °C for up to 10 minutes) are detailed in appendix D. The resulting material, termed reduced-graphite oxide (rGO), was several orders of magnitude more conductive than the starting material, clearly demonstrating the sensitivity of GO to heat. Such reduction of GO is known to occur at temperatures as low as 100 °C [84], imposing a limit on the thermal processing steps used during fabrication. In particular, the polymethylmethacrylate (PMMA) electron-beam resist is typically baked at approxi-

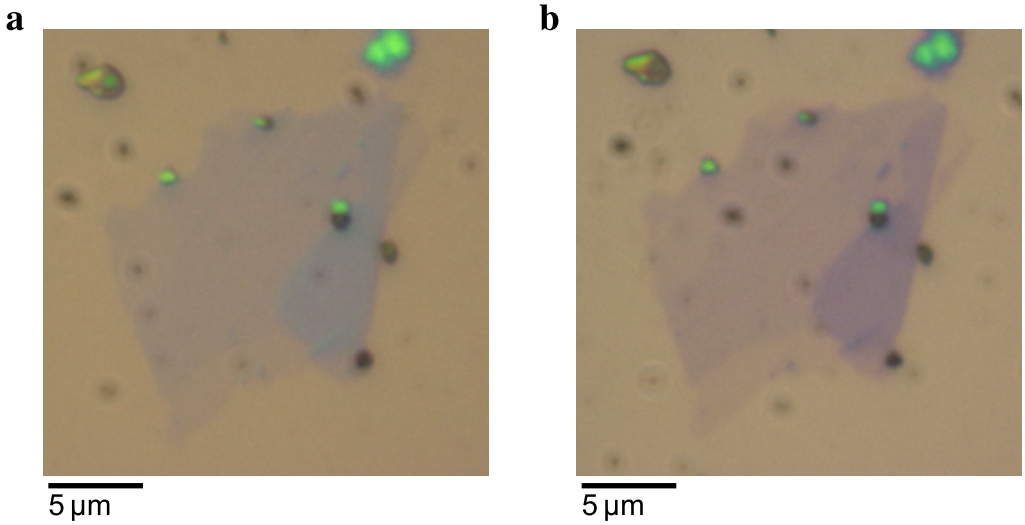


Figure 4.5: Optical micrograph under white light of a graphite oxide flake (a) immediately after deposition and (b) after a simulated electron-beam lithography process

mately 170 °C before exposure.

To further investigate the effect, graphite oxide flakes were deposited on several Si/SiO_2 substrates, which were then subjected to varying prebaking temperature and time before simulated electron-beam lithography. Optical micrographs of the flakes were recorded before and after using identical illumination, aperture, exposure, and color-balance settings. The observed color was then used as a qualitative measure of the resistivity [27, 85] and remaining oxidation of the partially reduced GO.

Images of a sample baked using the "standard" recipe, 170 °C for 15 minutes are shown in Figure 4.5. The graphite oxide flake changed color from blue-green to purple, indicating significant reduction. A subsequent test using 150 °C for 25 minutes showed less color change, while another at 120 °C for 25 minutes showed no detectable color change at all. Thus, special care was taken by using a "safe" recipe during fabrication of the G-GO FET samples to be measured. Given the results above, 115 °C for 15 minutes is considered to meet this requirement.

Another test was performed to determine whether the resolution of patterns created using the "safe" recipe would be significantly degraded. A 0.5 µm-scale test pattern was written at a range of electron doses and then metallized using thermal evaporation and liftoff. The best results were obtained when the dose was reduced approximately 10% (270 vs. 300 $\mu C/cm^2$ in this case) and were similar in quality to patterns made using the "standard" recipe.

4.3 Dielectric characterization

A gate dielectric material can be assessed in three main areas: insulation (leakage current), strength (breakdown electric field), and polarizability (relative dielectric constant κ). To measure these properties, G-GO capacitors and FETs were electrically characterized at room temperature in high-vacuum conditions and at low temperature in helium atmosphere using an Oxford variable temperature cryostat. Low-speed (DC) measurement techniques were used in both cases, as described in chapter 2. The samples tested were made using the "safe" low-temperature recipe described above unless otherwise noted.

4.3.1 Gate leakage current

First, a DC top-gate voltage V_{tg} was applied to the top-gates while measuring the leakage current I_{tg} collected by the graphene channels. Results for a relatively thin dielectric layer (thickness t_{go} = 4 nm) are shown in Figure 4.6a.[3] The gate current-voltage (I-V) curve is nonlinear, with a "knee" occurring at approximately 0.5 V in this particular device. Other devices with thicker graphite oxide layers show proportionally less current and wider flat regions, such that gate leakage is negligible in the thickest devices. In this device, the gate current became unstable when V_{tg} exceeded 0.5 V, indicating damage to the graphite oxide. At low temperature, the gate I-V is similar in shape and actually with slightly greater current, which can be attributed to minor additional damage caused by intervening measurements. Devices made with the low-temperature process typically show little change in gate leakage with temperature.

The effect of the processing parameters on gate leakage was investigated by fabricating G-GO capacitors using the "standard" recipe for PMMA-based electron-beam lithography. The low-bias gate leakage for such a device ($t_{go} \approx 4$ nm) is shown in Figure 4.6b. Indeed, at room temperature the gate resistance was only $\sim$500 kΩ, normalized to a 1 μm^2 area, which is approximately 100 times smaller than the resistance obtained using low-temperature processing. Interestingly, unlike the low-temperature-process devices, the gate leakage for this device strongly decreases with ambient temperature T, shown by the dashed line in Figure 4.6b, suggesting that transport through the gate

[3] At room temperature, the gate current density is significantly greater than for a typical SiO_2 film of similar thickness [86], but still below the threshold required to achieve transistor operation. The leakage could potentially be reduced further by using an improved synthesis recipe [87] to increase the oxidation level of the starting material or even-lower-temperature processing.

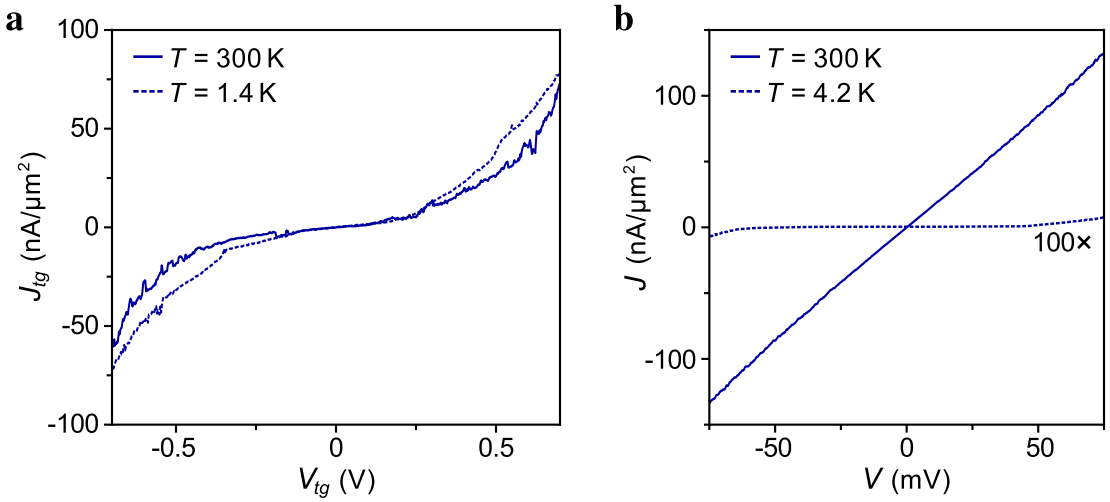


Figure 4.6: Gate-source I-V characteristics for two G-GO devices. (a) Data from a G-GO FET ($t_{go} = 4$ nm) recorded in helium atmosphere at room temperature (solid line) and 1.4 K (dashed line). The device was fabricated using a low-temperature process to bake the PMMA used for electron beam lithography. (b) Data from a G-GO capacitor ($t_{go} \approx 4$ nm) recorded at room temperature (solid line) and 4.2 K (dashed line). The latter curve has been multiplied by 100 for visibility. The device was fabricated using the "standard" recipe for PMMA-based electron-beam lithography.

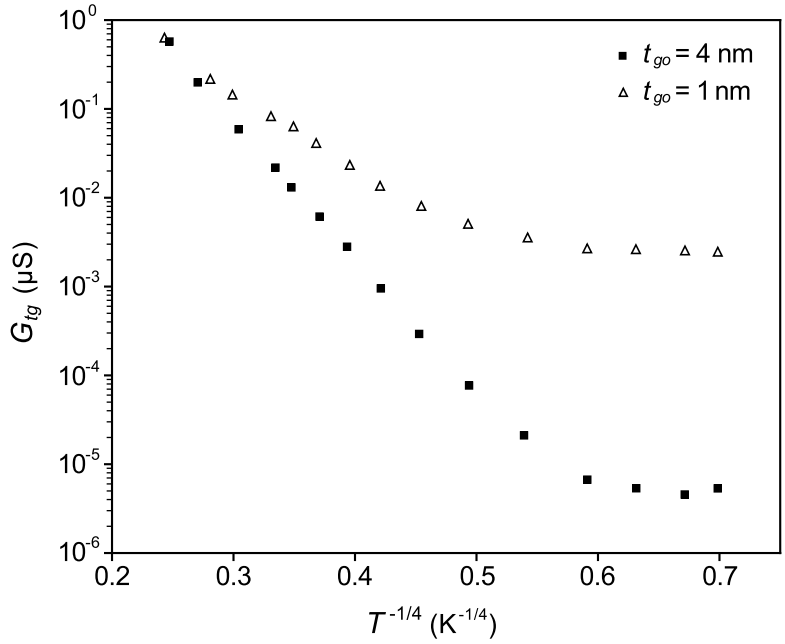


Figure 4.7: Gate conductance vs. temperature for two G-GO devices. Both devices were made using the "standard" recipe.

oxide occurs via a thermally activated process.

Gate conductance for two "standard"-recipe devices is plotted on a log scale against $T^{-1/4}$ in Figure 4.7. The linear decrease of $\ln(G_{tg})$ down to 10 K is consistent with a variable-range hopping model. The leakage current in this case is therefore unlikely to be related to the theoretical band gap [88] of GO in a straightforward way.

4.3.2 Breakdown electric field

Graphite oxide's breakdown electric field was measured by slowly increasing V_{tg} until I_{tg} showed a sharp jump, indicating irreversible damage. Data from a relatively thick device ($t_{go} \approx 35$ nm) measured at T = 10 K are shown in Figure 4.8. At 3.5 V the gate conductance became unstable, finally showing a large jump at 3.8 V. Thereafter the gate current followed a higher curve, identified by the downward pointing arrow, indicating dielectric breakdown. G-GO FETs can typically withstand stresses of greater than $\sim$1–3 $\times$ 10^8 V/m at room temperature, compared with $\sim 10^9$ V/m for bulk SiO_2 (e.g., [89]). With experience, one is able establish safe limits for the top-gate voltage for each device based on only the measured thickness (using atomic force microscopy) of the graphite oxide.

Approximately 10 devices were fabricated using the "standard" process and 20 devices using the low-temperature process. A significant fraction were found to have top-gate shorts, possibly caused by errors in lithography alignment or undetected pin-holes in the graphite oxide. Adopting

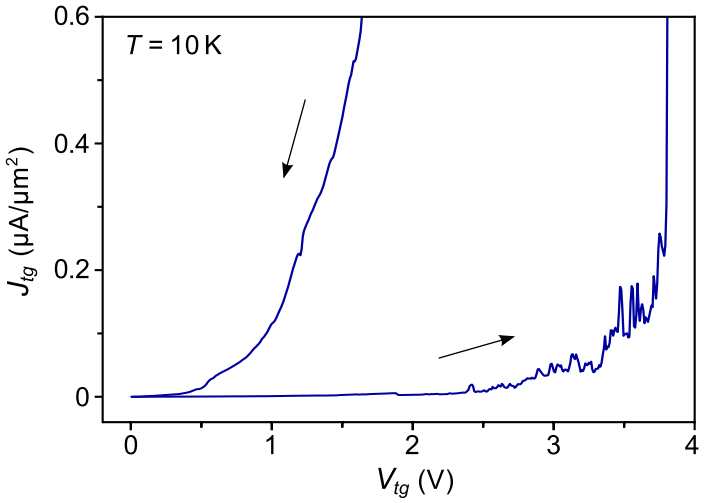


Figure 4.8: Dielectric breakdown of the GO layer in a G-GO FET

Table 4.1: Maximum applied gate stress for several G-GO transistors

t_{go} (nm)	V_{max} (V)	E_{max} (V/m)
4	1.3	3.1×10^8
∼10	1.0	1.0×10^8
18	4.2	2.3×10^8
35	3.5	1.0×10^8

a layer-transfer process to obtain more uniform graphite oxide or a self-aligned top-gate would probably improve the yield. Safe top-gate voltage limits were established as part of measuring each fully functional FET, and are listed in Table 4.1. The actual breakdown electric field is likely to be slightly higher than these values.

4.3.3 Estimation of dielectric constant

After establishing a safe range of top-gate voltage, the source-drain resistance of each G-GO FET was measured using low-frequency lock-in techniques while sweeping V_{tg} and back-gate voltage V_{bg}. The resulting resistance versus V_{tg} and V_{bg} data for a device with t_{go} = 18 nm, taken at T = 1.4 K with zero magnetic field, is shown in as a color-scale plot in Figure 4.9a. There is a pronounced resistance peak along the diagonal of the plot combined with a nearly horizontally oriented peak at approximately V_{bg} = 35 V, similar to that found in previous reports [67, 90–97]. These peaks can be understood by considering the schematic representation of the device shown in Figure 4.10. Regions *1* and *3* are not covered by the local top-gate and are therefore doped by only the back-gate. Region *2*, however, couples significantly to both gates. A simple model for the resistance is given as

$$R = \mu^{-1}\left(\frac{L_2/W}{C'_{bg}V_{bg} + C'_{tg}V_{tg}} + \frac{L_{1+3}/W}{C'_{bg}V_{bg}}\right) \tag{4.1}$$

where L_2 is the length of region *2*, L_{1+3} is the combined length of regions *1* and *3*, W is the sample width, and C'_{bg}, C'_{tg} are the back and top-gate capacitances per unit area, respectively; the mobility μ is assumed to be constant across all three regions and V_{bg} and V_{tg} are defined relative to the charge neutrality condition where all regions have zero doping.

Thus, sweeps at constant V_{bg}, shown in Figure 4.9b, have a single peak (attributable to the first term in equation 4.1) which shifts position with V_{bg}. Sweeps at constant V_{tg}, shown in Figure 4.9c,

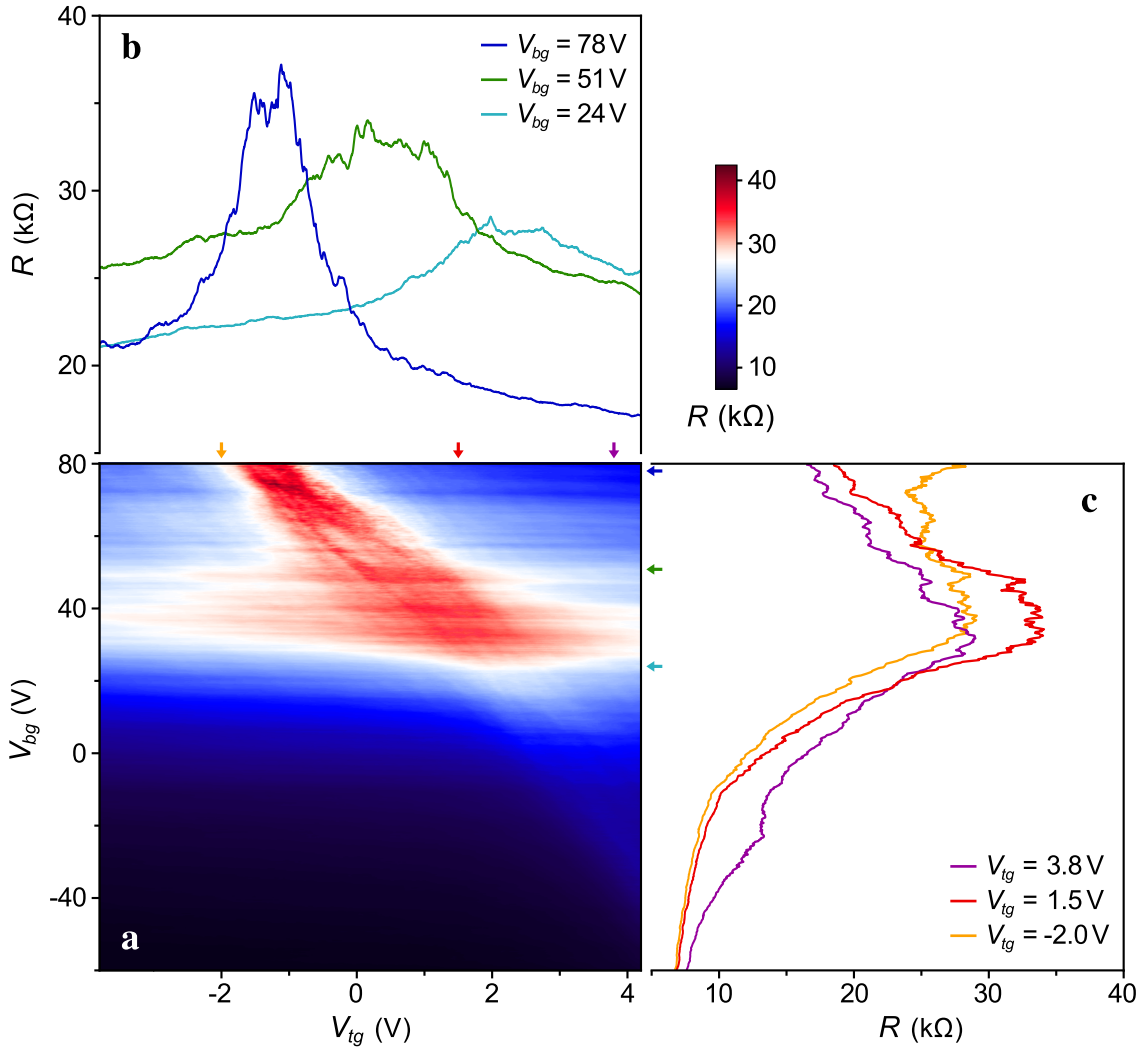


Figure 4.9: Two terminal resistance as a function of top-gate voltage and back-gate voltage at $B = 0$ and $T = 1.4$ K. (a) Color plot showing independent doping of channel regions *1–3* and *2*. (b) Horizontal data slices at constant back-gate voltage, with V_{bg} indicated by color-coded arrows placed at the upper edge of (a). (c) Vertical data slices at constant top-gate voltage, with V_{tg} indicated by color-coded arrows placed at the right edge of (a)

have one main peak corresponding to the Dirac point of regions *1* and *3* plus another peak which may occur above, below, or at the same value of V_{bg} depending on the contribution of the top-gate. The relative capacitance of the top and back-gates can be calculated by considering the denominator of the first term of equation 4.1, which is equal to the charge density in region *2*. When R is at a local maximum, as on the diagonal resistance peak in in Figure 4.9a, this charge density approaches zero, yielding the relationship

$$\frac{C'_{tg}}{C'_{bg}} = -\frac{V_{bg}}{V_{tg}}. \tag{4.2}$$

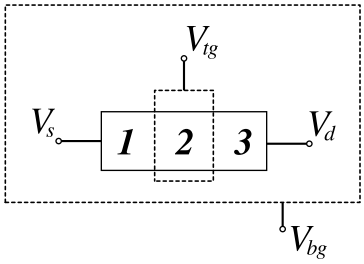


Figure 4.10: Schematic representation of a G-GO FET. The graphene channel, contacted at one end by source bias V_s and by drain bias V_d, is divided into three numbered regions defined by the edges of the top-gate.

Given t_{ox} = 290 nm, ϵ_{ox} = 3.9 ϵ_0, and t_{go} = 18 nm, the slope of this peak indicates graphite oxide's relative dielectric constant $\kappa \approx 4.3$. Thus, the top-gate may generate a carrier density of 5×10^{12} cm^{-2} while remaining below the breakdown field of graphite oxide mentioned above, which is sufficient for applications such as analog electronics [98]. Note that the quantum corrections to C'_{tg} and C'_{bg} were not included in this analysis.[4]

4.4 Carrier mobility

The carrier mobility of a FET may be estimated as follows: Consider the condition where V_{tg} is tuned give a uniform doping across the device, which is approximately the case for the red (V_{tg} = 1.5 V) curve in Figure 4.9c. In this case the mobility can be computed by fitting $R_s + L/W(\mu C'_{bg} V_{bg})^{-1}$ to the data, where R_s is the total series resistance and L is the total length. This device yields a value of approximately 700 cm^2/(V s)—comparable to, but somewhat less than the minimum mobilities observed in other studies of substrate-supported graphene devices: ≈1400 cm^2/(V s) by Moser et al. [46] and 2000 cm^2/(V s) by Tan et al. [99]. The origin of this relatively low value is not currently known.[5] One factor is that no postfabrication annealing step was done, which is typically required to achieve high mobility on SiO_2 substrates (see, for example, Morozov et al. [100] and Ishigami et al. [101]). Mobility might therefore be improved in future work by annealing the graphene in H_2 prior to GO deposition as an alternative to postfabrication

[4] Including the quantum capacitances would change the estimated κ by no more than a few percent. (It would increase because a smaller fraction of V_{tg} than V_{bg} contributes toward the electrostatic potential.)

[5] Control samples fabricated using the same process but lacking overlapping GO layers or top-gates showed similar mobility, suggesting that GO is not necessarily the limiting factor. Future experiments such as measuring the mobility before and after GO deposition will be necessary to clarify this issue.

annealing, which could compromise the GO dielectric. The conductance fluctuations visible in Figure 4.9 likely originate from a combination of disorder and phase-coherent transport within the graphene [102].

4.5 Future work

This work [103] demonstrates that graphite oxide shows promise as a layered dielectric, particularly because it is graphene's native oxide. Future work may improve the yield and reliability of the devices by using a layer transfer method [75] rather than random deposition to align GO with graphene sheets, or by using spray coating [33] to produce a continuous layer of GO flakes. Both approaches could be combined with oxygen plasma etching of windows in the GO accommodate source and drain contacts. While oxygen plasma is known to etch graphene, GO was found to be more vulnerable, suggesting that the process could be tuned to remove GO without damaging graphene.

It may be also be possible to selectively oxidize [77, 104–107] few-layer graphene to form an ultrathin insulating layer which is self-aligned to the underlying graphene, greatly simplifying the fabrication of graphene FETs. Graphite oxide may be used in novel nontransistor devices as well: It forms a versatile "insulating fabric" which could potentially be used as a tunnel barrier in both graphene and nongraphene devices. It might also enable the creation of ultrasharp p-n junctions with reduced fringing electric fields by allowing a local gate to be placed only a few nm away from graphene. Furthermore, it may be possible to replace the metal gate with yet another layer of graphene to form a thin, flexible, transparent all-carbon transistor, as illustrated in Figure 4.11.

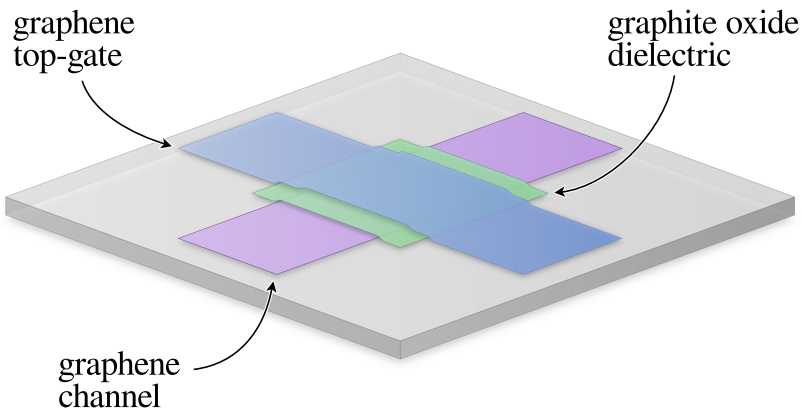


Figure 4.11: Illustration of a graphene-graphite oxide-graphene "sandwich" FET on a flexible substrate

Chapter 5

Data acquisition and measurement automation using *Mezurit 2*

5.1 Introduction

Automated test and measurement is an important area of development across fields from biology [108] to integrated circuit manufacturing [109]. "Big science" has evolved into "big data" [110], which necessarily entails "big measurement". In research the problem is amplified by the experimental nature of measurements; the worker often does not know *a priori* what region of parameter space is of interest. Instrumentation which offers immediate feedback may be ill-suited for repetitive, automated measurements. On the other hand, comprehensive hardware and software ecosystems such as National Instruments (NI) *LabVIEW* [111] or the MATLAB *Instrument Control Toolbox* [112] support automation but can require significant one-time programming investments.

Mezurit 2 is a software application designed to support electrical transport measurements common in the nanoscale electronics and low-temperature physics fields, similar in some ways to existing systems, both vendor-supplied [113] and user-generated [114, 115]. It strikes a balance between the above goals by providing simple channel definition, easy-to-use virtual instruments for common tasks, and immediate feedback in the form of real-time plotting. It supports automation by including a comprehensive scripting interface, which is accessible in real-time through a command-line terminal to facilitate the transition from manual to fully scripted operation. *Mezurit 2* is free software such that, unlike proprietary software, it can be freely used, copied, distributed, studied, and improved. These freedoms not only enhance productivity and promote collaboration; they are fundamental to good science [116].

5.2 General description

Mezurit 2 is a software application which sends commands to and receives data from data acquisition (DAQ) and general purpose interface bus (GPIB) hardware devices. The sequence of commands depends upon real-time inputs to its graphical user interface (GUI) or arbitrary scripts used for unattended experiments. The data received are converted, scaled, and combined according to a set of virtual channels.[1] The accumulated data are stored in memory, plotted for user inspection, and optionally saved to disk.

Mezurit 2 explicitly supports several common experimental paradigms through virtual instruments[1] called tools[1] (listed below). Each tool implements an algorithm for controlling or acquiring data from an experiment. Each algorithm depends on a collection of settings and controls which are exposed in a dedicated subsection of the GUI.

Acquisition	Records points[1] in real-time at speeds up to ∼1 kHz.
Scope	Records points for a finite time period at speeds up to ∼1 MHz.
Sweep	Linearly varies an output over a given range at a given ramp rate.
Trigger	Quickly responds to predefined events.
Terminal	Accepts commands to programmatically operate the other four tools.

Together, the first four tools cover the majority of common nanoelectronics experiments. The fifth tool—the *terminal*—is provided to cover the remainder. It runs the *Python* interpreter parallel to the main program to offer a set of text-based commands in the context of a full-featured programming language. The command set covers all controls and settings present in the GUI, plus several advanced "hidden" features. Commands may be typed-in directly or read from a script.

Up to 16 virtual channels can be defined using a flexible function-based entry system, enabling nonlinear scaling and arbitrary "channel math". The *acquisition*, *scope*, and *trigger* tools can access the full set of virtual channels, while each *sweep* tool associates with a single channel.

The GUI comprises a menu bar and a set of pages, only one of which is visible at a time. The menu bar contains controls to switch pages, save and load configuration profiles, view documentation, and access other miscellaneous features. There are three pages: one "Setup" page, which

[1] See Table 5.1 on page 53 for context-specific definition.

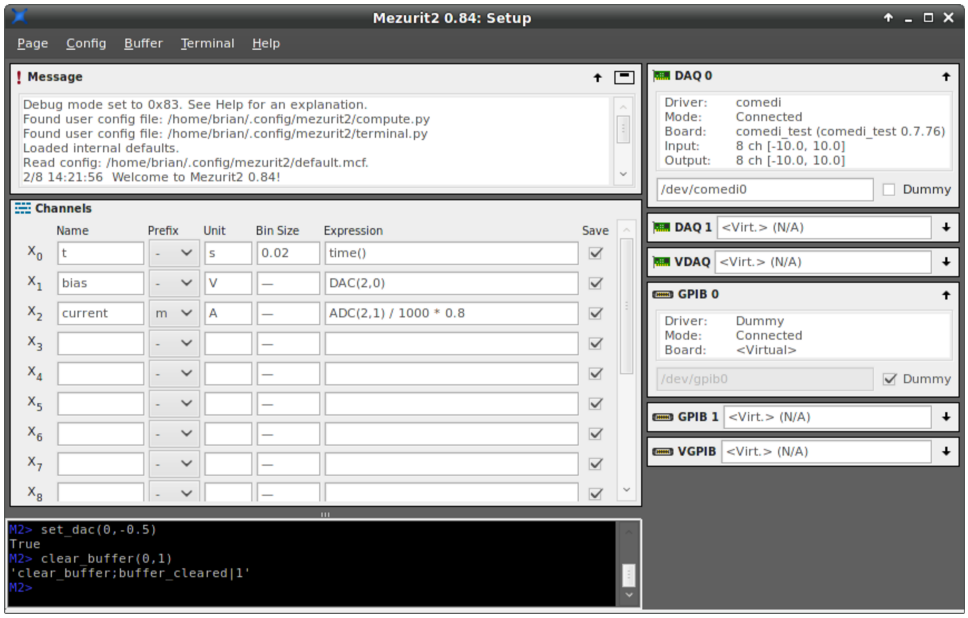


Figure 5.1: Screen capture of *Mezurit 2* running in setup mode

allows the user to define virtual channels and configure hardware, and two "Panel" pages, which expose the four main tools and a plot window.

Mezurit 2 operates in *setup mode*[2] while the "Setup" page is selected and in *panel mode*[2] while one of the "Panel" pages is selected. Screen captures showing the GUI running a demonstration experiment in these two states are shown in Figures 5.1 and 5.2, respectively. The *terminal* works in both modes and is therefore present on every page, though not all commands are available in both modes.

Mezurit 2 was inspired by a program called *Mezurit*[3] and as such the concepts of virtual channels, logging, and sweeps are present in both. The two applications share no code, however—*Mezurit 2* was written entirely from scratch by Brian Standley in consultation with Marc Bockrath. It comprises 10 854 lines of C and 323 lines of Python, not including comments and whitespace. It runs on GNU/Linux, Microsoft Windows XP, and Microsoft Windows 7 systems. The source code is online[4] and is released under the GNU General Public License [6].

A number of generic words and phrases are used in this chapter to describe concepts in the context of *Mezurit 2*. They are noted on first use and defined in Table 5.1.

[2] See Table 5.1 on page 53 for context-specific definition.
[3] Written by Marc Bockrath and David Cobden
[4] Available: `http://www.ugcs.caltech.edu/~mezurit2/`

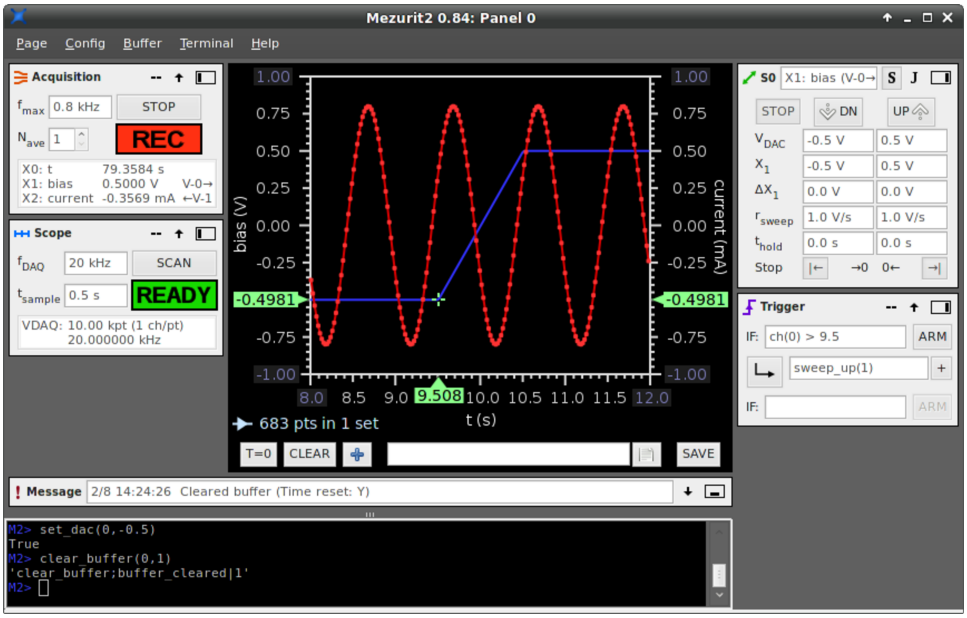


Figure 5.2: Screen capture of *Mezurit 2* running in panel mode

5.3 Operation

The operation of *Mezurit 2* is described in general and again in more detail as it relates to the problem of data acquisition.[5] A number of mathematical symbols are used in this chapter to describe the general (and subtle) behavior of the functionality provided. They are listed and defined in Table 5.2.

5.3.1 Overview

In a typical scenario, the user begins in setup mode by defining several virtual channels as functions of available physical channels[6] such as input and outputs provided by DAQ devices or GPIB-connected external instrument parameters. Switching to panel mode, the user chooses tool settings and then operates the measurement through their controls. Incoming data is saved to a buffer and plotted in real-time. The final dataset may then be saved to disk in a text-based format and loaded into many of the thousands of available data analysis programs.

As a shortcut, the user may also choose to load a previously saved configuration profile. Such profiles encompass every available setting—the program will be in the same state as the instant the

[5] Where it makes a difference, this section refers to version 0.91, released on 11 April 2012.
[6] See Table 5.1 for context-specific definition.

Table 5.1: List of context-specific terms

Term	Definition
virtual channel	A quantity defined by an arbitrary function of zero or more physical channels or other parameters, often corresponding to a physical quantity present in the device under test or elsewhere in the experiment
virtual instrument	A software-defined instrument which controls modular hardware instrumentation to accomplish a specific task
tool	A virtual instrument implemented in *Mezurit 2*, comprising a subsection of the GUI and an underlying model of operation
point	A set of real numbers corresponding to the values of the active virtual channels at a particular time
setup mode	The internal state of the application when the "Setup" page is selected, in which no acquisition loop is running
panel mode	The internal state of the application when a "Panel" page is selected, in which all tools are active
physical channel	An input or output port on a DAQ device or a quantity present in an external instrument
channel function	A Python function callable by a user-defined virtual channel function
parsing phase	The method of evaluation used to take an inventory of which channel functions are associated with a virtual channel
evaluation phase	The normal method of evaluated a virtual channel function purely to obtain a result
invertible channel	A virtual channel which meets certain requirements and can therefore be associated with a *sweep* tool
terminal command	A command, implemented as a Python function, used to effect some change in the operation of the measurement
trigger test	A Boolean Python expression used to determine when a *trigger* shall be executed
trigger command	A command similar to a terminal command executed in response to an predefined event

profile was saved, notwithstanding the contents of the data buffers and tool activity. *Mezurit 2* uses a text-based format called MCF (*Mezurit 2* configuration format) to store profiles.

The "Setup" page includes subsections to configure hardware along with the set of virtual channels. Up to two DAQ devices and two GPIB controllers can be connected at one time. *Mezurit 2* supports most NI DAQ devices and many others covered by the COMEDI open-source driver project [117]. NI GPIB controllers are supported along with others via the Linux-GPIB project.[7] In

[7] Website: `http://linux-gpib.sourceforge.net/`

Table 5.2: List of symbols

m	Index of a hardware unit, such as a DAQ device or GPIB controller
n, p	Generic channel indices; used for both virtual and physical channels
a	Primary address of a GPIB-enabled instrument
i	Index of a specific page, *sweep* tool, or *trigger* tool
ϕ	Text string such as a GPIB message or filename
f_n	Defining function of virtual channel n
X_n	Value of virtual channel n
X'_n	Target value to be impressed upon virtual channel n
t_l, t_u	Times of last acquired point and last sweep update, respectively
B_n	Sliding bin size setting for virtual channel n
ϕ_n	Python expression defining the body of virtual channel function f_n
ζ_{vc}	The set of valid virtual channel indices
T_i	Test expression for *trigger* i

addition, one virtual device of each class is always available for testing or other purposes.

Panel mode operation involves two main loops running in parallel. The "acquisition" loop is responsible for the low-level, time-sensitive tasks while the "interface" loop handles user input and output. Settings, control state, and data buffers are shared between the loops in a well-defined fashion, as described in section 5.5.

5.3.2 Virtual channel definition

Virtual channels correspond to quantities to be measured and include a description of how they are to be computed. The set of virtual channels defines the core of a *Mezurit 2* configuration profile, and their interpretation and evaluation is *Mezurit 2's* primary concern. Each virtual channel (with index n) comprises a name, International System of Units (SI) prefix, unit, bin size B_n,[8] and expression ϕ_n. The expression must be one line of Python code defining the function f_n used to compute the value of the virtual channel

$$X_n \equiv f_n\left(i_p, t_p, s_{vc,n}, S_{dac}, S_{adc}, S_{gpib}\right), \tag{5.1}$$

[8] The bin size parameter is described in section 5.3.3.

Table 5.3: Selected functions available for use in virtual channel definitions. The second column indicates the direction of information flow: ← for an input function, → for an output function, and ↔ for a bidirectional function.

Function		Description
$\mathrm{panel}()$	←	Index of current panel i_p, e.g., 0 for "Panel 0". (Returns −1 in setup mode.)
$\mathrm{time}()$	←	Time in seconds since the current panel was first started or its timer was reset
$\mathrm{ch}(n)$	←	Current value of virtual channel n, without SI prefix. For example, if X_2 = 3.4 mV, $\mathrm{ch}(2)$ would return 0.0034.
$\mathrm{DAC}(m, n)$	→	Value of analog output channel n on DAQ device m
$\mathrm{ADC}(m, n)$	←	Value of analog input channel n on DAQ device m
$\mathrm{A8648\_Freq}(m, a)$	↔	Agilent 8648 signal generator (with address a) frequency in Hz via GPIB controller m
$\mathrm{A8648\_Ampl}(m, a)$	↔	Agilent 8648 signal generator (with address a) amplitude in dBmW via GPIB controller m
$\mathrm{SR830\_SineOut}(m, a)$	↔	SRS SR830 lock-in amplifier (with address a) output amplitude in V via GPIB controller m
$\mathrm{SR830\_SensIn}(m, a)$	↔	SRS SR830 lock-in amplifier (with address a) input sensitivity in V/V via GPIB controller m

where i_p is the index of the current panel, t_p is the current panel's elapsed time, and $s_{vc,n} = \{X_p\} \mid p \in \zeta_x \cap \zeta_{vc}$ where $\zeta_x = \{0, 1, \ldots, n-1\}$ and ζ_{vc} is the set of valid virtual channel indices.[9] Sets S_{dac}, S_{adc}, and S_{gpib} contain the current values of all accessible physical channels of the output, input, or GPIB types, respectively. In practice, the details of fetching, converting, and scaling the arguments of f_n are abstracted away by channel functions.[10] There are 34 built-in channel functions, a selection of which are listed in Table 5.3. Additional functions may defined using the information in appendix E.

Virtual channels are called in two separate contexts, called the parsing[10] and evaluation[10] phases, and the behavior of their constituent channel functions depends on the current phase. The parsing phase occurs once each time the user switches from setup to panel mode (or between panels). Each expression ϕ_n is evaluated while channel function calls are analyzed to generate subsets s_{dac}, s_{adc},

[9] Because the set of virtual channel values is computed in order on each iteration of the acquisition loop, a virtual channel function should refer only to other channels of lesser index.

[10] See Table 5.1 on page 53 for context-specific definition.

and s_{gpib} containing the mentioned elements of their respective full sets.[11] This information is used to generate virtual channel metadata for display in the GUI and for other internal purposes. The expressions then compiled into Python function objects to speed up repeated computation. The evaluation phase occurs during each iteration of the acquisition loop when functions $\{f_n\}$ are called as needed to compute a new point $\{X_n\}$. Note that from here on it is implied that sets such as $\{X_n\}$ contain only members with indices $n \in \zeta_{vc}$, i.e., undefined virtual channels are simply ignored in panel mode.

Channel functions may be split into three categories according to the direction of information flow: input, output, and bidirectional. Input functions are the simplest to understand—when the function is called a value is fetched from internal variables (i_p or t_p) or the analog input buffer (s_{adc}), which are updated continuously in panel mode. Output functions are more complicated —they behave like input functions during the evaluation phase by returning the last known value of the physical channel in question. In addition, during the parsing phase they flag the calling virtual channel as potentially usable by the *sweep* tool. Such virtual channels may be considered invertible,[12] and are described in more detail below. Finally, bidirectional functions are similar to output functions, but the known value is periodically queried from an external instrument. Many functions encapsulating GPIB commands fall into this category.

Virtual channels defined by a linear function f_n of exactly one output or bidirectional channel function are considered invertible, meaning they can be controlled by *sweep* tools or updated by a terminal command.[12] When a new target value X'_n is set, the inverse of f_n is used to compute the necessary output $x'_n = f_n^{-1}(X'_n)$. The result is then immediately impressed upon the applicable analog output (DAC) channel or transferred to the appropriate instrument over GPIB. While it would be possible to use an iterative algorithm such as Newton's method to find x'_n given an arbitrary (but well-behaved) f_n, *Mezurit 2* is currently limited to linear f_n where inversion is concerned. This restriction enables f_n^{-1} to be worked-out during the parsing phase and efficiently computed during the evaluation phase.

[11] Subsets may change during the evaluation phase for certain, exotic, f_n, which is technically supported but not recommended. Typically, the arguments of channel functions are constant or depend on only i_p.

[12] See Table 5.1 on page 53 for context-specific definition.

5.3.3 Logging and scanning

Mezurit 2 acquires data in two modes: logging and scanning. The former mode is controlled by the *acquisition* tool and the latter by the *scope* tool. In logging mode points are acquired continuously at a variable rate and recorded to a data buffer only if certain conditions are met. Scanning, by contrast, configures the DAQ hardware to acquire points at a constant rate for a limited time and records every point.

The *acquisition* tool includes several settings which affect logging mode operation plus a read-out which displays the set of virtual channels and their current values. The f_{max} setting specifies the maximum acquisition rate, subject to the limits of the hardware and operating system.[13] Under normal circumstances a new point will be acquired every $1/f_{max}$-time period. Machines equipped with older single-core processors may experience occasional longer delays due to the overhead of updating the GUI. In practice, rates of 0.6–1.0 kHz (typically limited by DAQ hardware) are achievable on most systems.[14] The N_{ave} setting specifies the number of raw points used to generate a moving average. While the same filtering can be applied offline, it can be useful to prefilter noisy inputs to enhance the clarity of plots or make triggering more reliable.

At this stage it is important to emphasize the distinction between *acquired* and *recorded* points: An acquired point is the raw $\{X_n\}$ generated on each iteration of the acquisition loop.[15] In becomes a recorded point when it is saved to the data buffer, which occurs only when the point meets the binning criterion. *Mezurit 2* implements a sliding binning scheme as follows: An acquired point may be recorded when decision variable d is true, as defined by

$$\bar{d} = |X_n - X_{n0}| < B_n \,\forall\, n \in \zeta_{vc}, \tag{5.2}$$

where X_{n0} is the last recorded value. Thus, a point will be recorded when any virtual channel's value excurs more than its bin size away from the last recorded value. Setting any B_n to zero causes *Mezurit 2* to recored every acquired point. Leaving a bin size undefined is equivalent to setting B_n to ∞ such that virtual channel n will never trigger a recorded point.

[13] Some Microsoft Windows-based computers lack the ability to generate accurate sub-1 ms time delays. In those cases, *Mezurit 2* falls back on a less efficient loop-based timing system.

[14] 3–4 kHz has been demonstrated using simulated DAQ hardware.

[15] Averaging is applied between the computation of $\{f_n\}$ and the updating of $\{X_n\}$, such that the *acquisition* tool readout and results of trigger tests (see section 5.3.5) reflect the averaging.

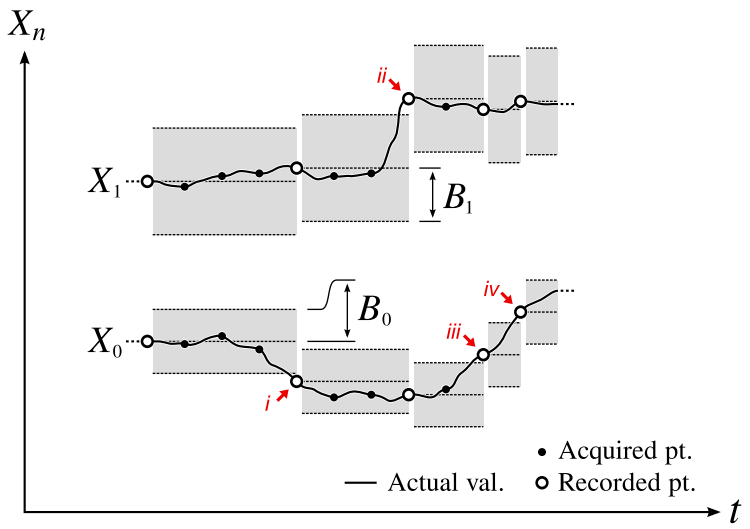


Figure 5.3: Example illustrating the sliding binning scheme used in *Mezurit 2*. The numbered red arrows indicate binning events.

Several examples of binning events (marked by red arrows) are illustrated in Figure 5.3. The values X_0 and X_1 remain within their respective bins through the first three acquired points after the initial recorded point, at which point the overall decrease of X_0 exceeds B_0 (event *i*). A new point is recorded and the bins for both virtual channels are adjusted. Event *ii* occurs when X_1 rises sharply out of its bin, demonstrating that if a channel rises quickly relative to characteristic slope $B_n f_{max}$, then the difference between successive recorded values may be noticeably larger than B_n. Events *ii* and *iv* show how a smooth rise generates a smooth staircase of recorded values. The binning system's primary function is to limit the quantity of data taken when the experiment is idle or noisy. If desired, a perfectly regular sequence of X_n may be generated in some circumstances using the *sweep* tool, described in the next section.

The *scope* tool provides an interface for the scanning mode of acquisition. When a scan begins the DAQ hardware is instructed to take samples at a fixed rate f_{DAQ} for a fixed length of time t_{sample}, transferring the raw ADC data to a temporary buffer. The samples are then converted, scaled, and used to compute $\{X_n\}_j$ at each time index $j \in \{0, 1, \ldots, t_{sample}\, f_{DAQ} - 1\}$. A virtual channel must meet certain conditions to be part of a scan: 1) f_n must depend on only i_p, t_p, and/or s_{adc} and 2) if multiple ADC channels are referenced, they must belong to the same DAQ device.[16]

[16] Because scans are timed by the DAQ device's internal clock, samples from different devices cannot be reliably "matched-up" to compute X_n even when the f_{DAQ} are nominally equal.

Unscannable channels will be computed once based on conditions immediately after the scan and then copied to every point as constants.

The advantages of scanning mode are potentially much faster acquisition (up to 1 MHz depending on DAQ hardware) and periodic sample times. Scans can be started and stopped using GUI controls or commands sent from the *trigger* or *terminal* tools. The command method enables precise synchronization with external hardware using the $\mathrm{fire\_scope\_pulse}(n, X'_n)$ terminal command, which sends a pulse on virtual channel n immediately before the scan.

5.3.4 Sweeping

Electronics experiments, unlike, e.g., seismic monitoring, rarely require only passive data acquisition. Characterizing a time-independent system depends on recording its response over a region of N-dimensional parameter space. Assuming the response varies smoothly as a function of each parameter, one might divide the space into discrete volume elements and then devise a path passing through every element's location. *Mezurit 2* provides multiple *sweep* tools to accommodate any such path, provided the parameters can be expressed as invertible virtual channels. One dimensional paths are sometimes called *sweeps* (hence the name of the tool), two dimensional paths, *megasweeps*, and three dimensional paths, *gigasweeps*. Megasweeps and gigasweeps can be automated using the *terminal*, as described in section 5.4.1.

Each *sweep* tool associates with a single invertible virtual channel, ramping its value linearly up or down at a constant rate. Upper and lower limits can be set in terms of X_n or the equivalent physical output value x_n. The ramp rate r_{sweep} is expressed in the units of X_n and can be set independently for each direction. Upon reaching a limit, the sweep either stops or changes direction after a configurable hold time. The sweep can also be set to automatically stop at zero.

Measurements of even time-independent systems can themselves be time-dependent due to parasitic capacitance, inductance, and/or other factors. For this reason it is useful to consider how, exactly, parameters vary with time between recorded points. The sweeping operation is therefore described in detail for both the simplest case where step size $\Delta X_n = 0$ and the more-complicated "stepping mode" where $\Delta X_n > 0$.

Sweep tools operate during only logging-mode operation (all active sweeps are automatically

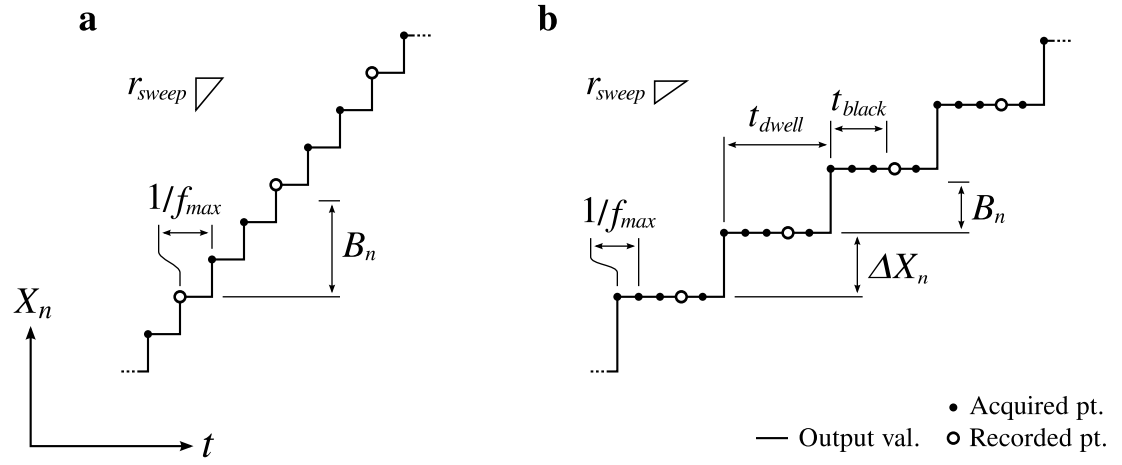


Figure 5.4: Output value and acquired/recorded points while sweeping a single virtual channel. (a) The simplest case in which $\Delta X_n = 0$ and $B_n \gg r_{sweep}/f_{max}$. (b) "Stepping" mode, where $0 < B_n < \Delta X_n$ and $t_{dwell} \gg 1/f_{max}$. In both cases it is implied that $N_{ave} = 0$ and the respective bin sizes of the other virtual channels are not a factor during the sweep.

stopped before scanning) and are therefore subject to the uncertainties of the acquisition loop. Thus, in the simplest case the actual sweep output

$$X_n(t) = (t_l - t_0)r_{sweep} + X_{n,0}, \tag{5.3}$$

where $(t_0, X_{n,0})$ is the initial coordinate, t_l is the time of the most recent iteration of the acquisition loop, and r_{sweep} is the direction-dependent rate set for the *sweep* tool associated with virtual channel n. Under optimal conditions, the plot of t_l vs. t therefore resembles a regular staircase with an average slope of unity and $X_n(t)$ resembles a similar staircase, but with average slope r_{sweep}, shown in Figure 5.4a. The bin size B_n of a swept channel affects how often an updated X_n triggers a recorded point. In the example shown in Figure 5.4a B_n is between two and three times the characteristic step size r_{sweep}/f_{max}, so every third point is recorded.

Given that the time between updates may vary slightly from $1/f_{max}$, the step height must vary as well to maintain the requested rate. The form of equation 5.3, however, guarantees that variance in the step heights does not cause any long-term error. If desired, however, the step size can be fixed by setting ΔX_n to a nonzero value, putting the *sweep* tool in "stepping" mode. The step height is

then converted into an equivalent dwell time

$$t_{dwell} = \Delta X_n / r_{sweep}. \tag{5.4}$$

The GUI provides settings for both ΔX_n and t_{dwell}, updating both simultaneously based on equation 5.4. Sweep updates prioritize ΔX_n over t_{dwell}, however, such that the following conditions are true:

$$X_n = k(\Delta X_n) + X_{n,0} \mid k = \lfloor (t_l - t_0)/t_{dwell} \rfloor \tag{5.5}$$

$$\lim_{f_{max} \to \infty} \Delta t_u = t_{dwell} \tag{5.6}$$

Thus, the time between updates Δt_u merely approaches t_{dwell}. The difference between the two can be minimized, however, by setting $f_{max} \gg 1/t_{dwell}$.

One example of a time-dependent measurement effect is the settling time some instruments require to readjust to a new equilibrium state of the system. A settling period can be accommodated using the t_{black} setting, which suspends recording (but not acquisition) after each sweep update for a specified length of time up to t_{dwell}.[17] Stepping mode with finite t_{black} is illustrated in Figure 5.4b. A new point is recorded near the middle of each plateau of $X_n(t)$, not at the leading edge as would normally be the case. The actual time delay between sweep updates and recorded points may be slightly longer than t_{black}—up to $t_{black} + 1/f_{max}$. In general, *sweep* tools run more smoothly when f_{max} is significantly faster than other time scales of the system, such as r_{sweep}, t_{dwell}, and t_{black}, and the recording rate is controlled through binning.

Note that for both cases shown in Figure 5.4, it is implied that none of the other virtual channels' bin sizes is relevant. A bin size B_n is considered relevant if it is defined and smaller than the expected variation of X_n over the course of the sweep, which depends on the details of the experiment.

Finally, some experiments require simultaneously varying two or more parameters, effectively plotting an angled path through the parameter space. For example, the out-of-plane electric field in a dual-gated field-effect transistor may be tuned at constant doping by sweeping one gate voltage

[17] In the GUI, the blackout time is expressed as a percentage of t_{dwell}, but is defined here as a time variable: $t_{black} = t_{dwell} \times$ [% blackout].

up and the other down. This can be accomplished by referencing one invertible virtual channel to another using the $\mathrm{set\_follower}(n_l, n_f, \phi)$ terminal command. The "follower" (channel n_f) will then be set any time the "leader" (channel n_l) is set, according to

$$X'_{n_f} = f(X'_{n_l}), \tag{5.7}$$

where X'_{n_l} and X'_{n_f} are the target values of the leader and follower, respectively, and f is an arbitrary function defined by Python expression ϕ.[18]

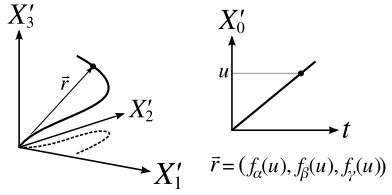
5.3.5 Synchronization

Mezurit 2 provides several mechanisms to synchronize internal and external events. The most general are the *trigger* tools, which execute a sequence of commands in response to an arbitrary event. For example, a *trigger* can automatically start a scan when a 5 V TTL-compatible sync signal is received from an external function generator. Each *trigger* tool i contains a Boolean expression T_i called a trigger test[19] which is evaluated on every iteration of the acquisition loop, provided the *trigger* has been "armed". The test expression shares the same namespace with the virtual channels and can therefore access all the same functions (most commonly $\mathrm{time}()$ and $ch(n)$). If and when T_i evaluates to true an arbitrary sequence of trigger commands[19] is executed and the *trigger* is disarmed. The available commands are a subset of the terminal commands, described below.[20]

The *terminal* provides other synchronization mechanisms such as sweep events, scan events, and trigger signals. Sweep events allow a script to wait for a *sweep* tool to reach an end or zero point

[18] This approach can be taken one step further by expressing two or more follower channels as a function of a free parameter represented by a "dummy" channel. For example, to trace a three dimensional path (illustration below) configure four virtual channels

$$\begin{aligned} X_0 &= \mathrm{DAC}(2,0) \\ X_1 &= f_1(\dots) & X'_1 &= f_\alpha(X'_0) \\ X_2 &= f_2(\dots) & X'_2 &= f_\beta(X'_0) \\ X_3 &= f_3(\dots) & X'_3 &= f_\gamma(X'_0) \end{aligned}$$



and then linearly sweep virtual channel 0, which has no physical significance, to acquire data along the path specified by the follower functions f_α, f_β, and f_γ.

[19] See Table 5.1 on page 53 for context-specific definition.

[20] Though nearly all trigger commands are syntactically identical to equivalent terminal commands, they are implemented internally in different ways. Most crucially, they execute directly inside the acquisition loop and therefore must return promptly.

using the $\mathrm{catch\_sweep}(\phi, n)$ command, which blocks until a certain event to occurs while sweeping virtual channel n. A similar pair of functions exist for the *scope* tool: $\mathrm{catch\_scan\_start}()$ and $\mathrm{catch\_scan\_finish}()$. Finally, trigger signals allow *trigger* tools to communicate with the *terminal*. A *trigger* may send a signal with the $\mathrm{emit\_signal}(\phi)$ command, which is "caught" by the terminal command $\mathrm{catch\_signal}(\phi)$.

5.4 Scripting

The comprehensive scripting interface available through the *terminal* closely mirrors the set of virtual instruments found in the GUI. This makes transitioning from manual to automated operation as straightforward as possible. *Mezurit 2* includes many features which fall into three categories: 1) controls exposed by buttons and menu items, 2) settings, and 3) hidden functions. *Every* control that is logically scriptable has a dedicated counterpart terminal command. In contrast, settings are read and/or written using a generalized pair of advanced commands, described below.

Commands may be directly typed into the *terminal* or placed into scripts to be called from the *terminal*. The *terminal* is implemented as an embedded display running a *Python* interpreter in a separate process from the main program. This design protects the integrity of the time-sensitive data acquisition process from untested scripts, which may hang or fail. The interpreter automatically restarts after a crash and can be manually restarted in the GUI.

There are 45 built-in terminal commands covering all three categories. A representative selection of those which directly correspond to "clickable" controls in the GUI is given in Table 5.4. Additional commands may be defined using the information in appendix E.

5.4.1 Advanced commands

A selection of advanced commands used to read or write settings or access "hidden" features is given in Table 5.5. The commands $\mathrm{get\_var}(\phi)$ and $\mathrm{set\_var}(\phi)$ refer to settings using the same key-value pairs used in MCF files. Thus, finding the appropriate string ϕ is as simple as visually inspecting any MCF file. Changing a setting from the *terminal* activates the same internal update procedure as the manual method, meaning that the new value cannot create any internal inconsistency in the operation of the tools.

Table 5.4: Selected basic terminal commands used to access controls available in the GUI. Commands marked with 'S' are usable in setup mode, 'P' in panel mode. Those also usable as trigger commands are marked 'Tr.'

Function		Description
$\mathrm{load\_config\_file}(\phi)$	S	Load MCF file ϕ
$\mathrm{save\_config\_file}(\phi)$	S/P	Save current configuration to file ϕ
$\mathrm{set\_recording}(b)$	P/Tr	Turn recording on or off according to b
$\mathrm{save\_data}(\phi)$	P/Tr	Save the data buffer to file ϕ
$\mathrm{clear\_buffer}(b, c)$	P	Clear the data buffer after confirmation from the user (if $b = 1$) and reset the time (if $c = 1$)
$\mathrm{set\_dac}(n, X'_n)$	P/Tr	Set (invertible) virtual channel n to target value X'_n
$\mathrm{sweep\_up}(n)$, $\mathrm{sweep\_down}(n)$, $\mathrm{sweep\_stop}(n)$	P/Tr	Control the *sweep* tool associated with virtual channel n[a]
$\mathrm{fire\_scope}()$, $\mathrm{cancel\_scope}()$	P/Tr	Start or stop a scan
$\mathrm{arm\_trigger}(i)$, $\mathrm{disarm\_trigger}(i)$	P/Tr[b]	Arm or disarm *trigger* tool with index i[a]

[a] Note that the *sweep*-based commands take virtual channel numbers n as arguments while *trigger*-based commands require tool index i. A utility function $\mathrm{get\_sweep\_id}(n)$ is provided to find *sweep* index i given virtual channel n when needed.

[b] Setting a *trigger* to arm a *trigger* can be useful for multistage events, or to automatically re-arm.

The $\mathrm{gpib}(m, a, \phi)$ command is of particular importance because it allows arbitrary messages to be sent over GPIB without interfering with any GPIB-based virtual channels. Message ϕ is transferred to the main program and then inserted into the ongoing sequence of messages sent and received.

One common application of the scripting interface is the automation of megasweeps and gigasweeps. The former takes the form of a loop in which each iteration increments one parameter using $\mathrm{set\_dac}(p, X'_p)$ and then sweeps another parameter using $\mathrm{sweep\_up}(p)$ followed by $\mathrm{catch\_sweep}(\text{'max'}, p)$. The data may be saved after each sweep using $\mathrm{save\_data}(\phi)$ or simply saved manually at the end of the megasweep. While composing and calling a script may be more time-consuming than simply configuring a hypothetical *megasweep* tool, there is much more flexibility in this approach. In addition, it allows a separation of mechanism, i.e., basic tools and terminal commands, from policy, which would necessarily be imposed by such a tool's model of operation. A more involved example script is given in appendix E.

Table 5.5: Selected advanced terminal commands including those which expose hidden features. Commands marked with 'S' can be used in setup mode, 'P' in panel mode.

Function		Description
$\mathrm{set\_follower}(n_l, n_f, \phi)$	P	Defines a leader-follower relationship between virtual channels n_l and n_f defined by Python expression ϕ
$\mathrm{catch\_sweep}(\phi, n)$	P	Waits for event ϕ to occur while sweeping virtual channel n
$\mathrm{catch\_signal}(\phi)$	P	Waits for signal ϕ to be emitted by a trigger command
$\mathrm{gpib}(m, a, \phi)$	S/P	Sends message ϕ to GPIB address a via controller m, returns reply
$\mathrm{set\_var}(\phi)$	S/P	Sets configurable variable according to line ϕ of the form "key=value"
$\mathrm{get\_var}(\phi)$	S/P	Returns the value of variable ϕ as a string

5.5 Architecture

This section is intended to give an overview of *Mezurit 2's* architecture. First, the overall structure of the source code is described, followed by an account of how the acquisition and interface loops are implemented as separate threads for performance. Several advanced programming concepts including remote procedure calls (RPCs) and callback functions with pseudoclosures are employed to make the source code amenable to further development while allowing the necessary complexity of the design. These are described briefly, with examples, in the final section.

5.5.1 Modularity

Mezurit 2 is designed to be as easy as possible to understand at the source-code level given the complex nature of its task. Toward this end, its functionality is divided into modules, which were refactored continuously during development. Modularity in any given computer program can take multiple forms. For example, functions may be divided by level of abstraction or practical features. These two approaches in particular are incompatible for *Mezurit 2*, given that each tool (a practical feature) encompasses GUI controls, configuration settings, and real-time logic to implement its model of operation. Thus, a hybrid approach, diagrammed in Figure 5.5, was taken where higher-level functions are organized by tool (including pseudotools such as the message window, data buffer, and plot display). All lower-level functions, however, are grouped together given their highly interrelated nature.

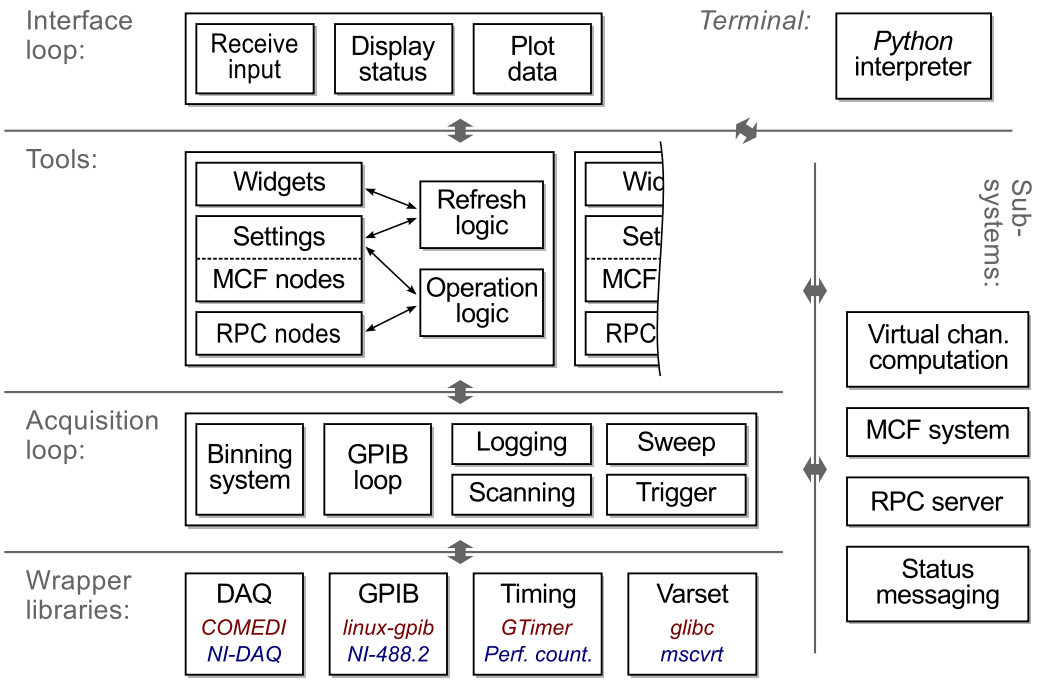


Figure 5.5: Block diagram of source code-level modularity. Blocks are grouped by abstraction level —from the user interface at the top to hardware access at the bottom. Here, the "tools" group includes the *acquisition*, *scope*, *sweep*, and *trigger* tools, as well as pseudotools such as the message window, data buffer, and plot display. System libraries are color-coded by platform: red for GNU/Linux, blue for Microsoft Windows.

The four main tools—*acquisition*, *scope*, *sweep*, and *trigger*—are straightforward to understand. Each is built around a set of configuration settings and GUI controls (implemented as *GTK+* widgets[21]) along with logic to handle updates and events. Pseudotools are similar at the source code level, but are typically shared between the main tools. For example, the *acquisition* and *scope* tools share access to the data buffer.

The code required to implement any single tool is reduced by the presence of several subsystems. One such system is the *GTK+* event loop which can be instructed to connect user input events, such as clicking on a button or updating a setting, with callback functions. This way tool logic may be encapsulated and placed near the tool's data structures rather than in the acquisition or interface loops directly.

A similar scheme is used in the MCF subsystem used to save and load settings and to implement terminal and trigger commands as RPCs. Each configuration setting is associated with an MCF node comprising a key string, type, default value, and callback function. When an MCF file is loaded,

[21] The *GIMP Toolkit*, available: `http://www.gtk.org/`

each line is matched to a unique node using the key string and a value is extracted according to the specified type. The callback function is then called to safely update the variable and perform needed housekeeping tasks. The RPC system is similar but uses text-based commands which are received over a socket from the *terminal* process. Each command is matched with a callback, effectively allowing scripts to remotely "call" functions in the main process. Trigger commands are processed by a separate instance of the *same RPC system*, even though they actually originate in the main process.

5.5.2 Multithreading

> *In the data acquisition system, the users are represented by two separate but equally important threads: DAQ, which records the points, and GUI, which plots the curves. These are their stories.*

Multithreaded applications are qualitatively more difficult to compose and debug [118]. In the case of *Mezurit 2*, however, the advantages greatly outweigh the disadvantages, especially given the range of time scales involved (Table 5.6). When running on multicore machines, it can acquire data, communicate over GPIB, and process user input simultaneously. The acquisition loop may thus run unimpeded by other computational or input/output (I/O) tasks. Single-core machines, too, benefit from multithreading in the form of simplified integration of blocking I/O operations. For example, GPIB communication is relegated to a separate thread where slow-to-respond instruments may be accommodated without resorting to complex asynchronous I/O.

The main process comprises three threads, though only one (GUI) runs in setup mode: "DAQ", which runs the acquisition loop; "GPIB", which sends GPIB messages, waits, and receives replies; and "GUI", which runs the interface loop. Upon switching to panel mode, the GUI thread initializes and forks a new DAQ thread to handle time-sensitive operation, which in turn initializes and forks a new GPIB thread. All three threads run simultaneously from that point until the user leaves the current panel. Configuration settings and data buffers are safely shared between threads using mutex-style locks. Choosing the appropriate level of granularity for the locking scheme can be challenging. For the most part, *Mezurit 2* protects data at the tool level. For example, while the DAQ thread running a critical section to update the sweeps, the *sweep* tools are protected but the *trigger* tools remain available for configuration.

Table 5.6: Time scales present in the operation of *Mezurit 2*

DAQ THREAD	
Default sampling period in logging mode (f_{max})	1.25 ms
Typical sampling period in scanning mode	10 µs
Polling period for terminal commands	2.5 ms
GPIB THREAD	
Minimum time between sending of messages	20 ms
GUI THREAD	
Polling period for *GTK+* events	8.3 ms
Time between updates to the virtual channel readout	14 ms
Time between updates to the data buffer status	24 ms
Time between updates to the *scope* progress bar	91 ms

The *terminal* is implemented by as separate process embedded in the main GUI using *VTE*.[22] *VTE* transmits keystrokes to its child process and displays output received. If the child process should crash, it may be automatically restarted without negatively affecting the main process.

A full account of the shared data and synchronized interactions between threads is not given here, but a schematic of the general structure is shown in Figure 5.6. Branching points are shaded grey, actions are blue, and data buffers are white with black outlines. Control flow is indicated by solid lines and data flows are dashed lines, whether the endpoints are branching points, actions, or buffers. Protected transfers to or from shared buffers are marked with a padlock symbol. A few key areas are highlighted:

- Both the DAQ and GUI threads maintain private buffers to hold recently acquired points. A third, shared buffer is used to facilitate the propagation of data between the two with minimal overlap. Thus, the DAQ thread *never* needs to wait to update X_n and *almost never* waits to transfer a point to the shared buffer, because the GUI thread keeps a copy for use in the potentially slow process of updating the *acquisition* tool readout.
- The RPC system is shared between threads even though all RPC commands come into the system through the DAQ thread. This minimizes the response time of acquisition-related commands while dispatching slower, interface-related commands to the GUI thread.
- The green-shaded actions may invoke callback functions which affect the operation of multiple threads (while holding the appropriate locks, of course). For clarity, such interactions are not shown in the figure and are in fact not explicitly mentioned in the acquisition loop's code. Some of the subtleties of these events are explained via examples in the next section.

[22] The *Virtual Terminal Emulator* library, documentation: `http://developer.gnome.org/vte/0.30/`

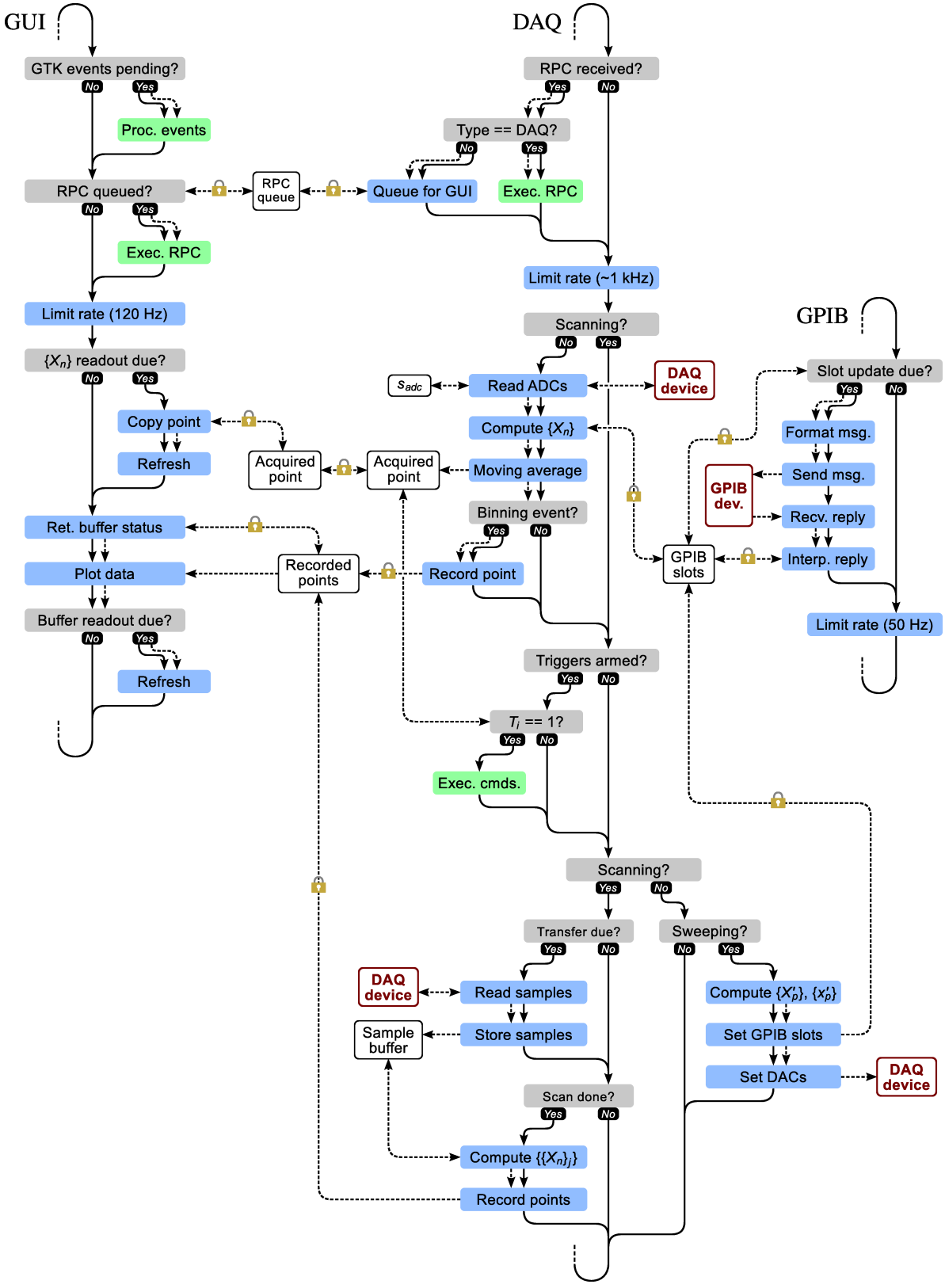


Figure 5.6: Schematic of threads running in panel mode and the interactions between them

5.5.3 Callbacks and pseudoclosures

Each tool contains unique logic to handle updates to its controls and settings, which is expressed as a collection of callback functions. Callbacks are implemented in the C language using function pointers [119], which lack the advanced features of truly functional languages such as lambdas and closures. Closures in particular would be useful to have because they allow a callback to access nonlocal variables without resorting to global variables. Toward that end, a limited pseudoclosure construct was implemented to augment the *GTK+* callback registration process. When a callback function is registered, a set of additional values and variable references is captured and stored. Later, when the pseudoclosure is invoked, the function is called with access to the captured data. The same construct is used in the MCF and RPC subsystems.

An example of a pseudoclosure in action is shown in Figure 5.7. Each *sweep* tool includes a ramp rate setting (actually, two) which affects how the acquisition loop updates its associated virtual channel during sweeping. This setting is stored as an internal variable "rate_var" and exposed to the user as the r_{sweep} *GTK+* widget. A callback function "setvar_cb" is defined to handle changes to the *sweep* tool's settings. At program startup, a pseudoclosure is created (step 1) comprising an event name and references to the callback, variable, and widget. When the user updates the setting in the GUI (step 2), an event is queued which is then matched to the pseudoclosure by the "gtk_main_iteration" function (step 3, corresponding to the extreme upper left corner of Figure 5.6). The pseudoclosure is invoked (step 4), fetching the value from the appropriate widget (step 5) and safely updating the variable (steps 6–8). The DAQ thread will now sweep at the new rate (step 9).

The scenario introduced above is extended to show how the MCF and RPC subsystems function similarly in Figure 5.8. Each *sweep* tool also includes a MCF node with key "sweep_rate" which lets the MCF system know about the internal variable "rate_var". The terminal command $\mathrm{set\_var}(\phi)$ provides remote access to the MCF subsystem through the RPC subsystem, such that a script can update the ramp rate in much the same way the user did using the r_{sweep} widget. A callback function "setvar_mcf" is defined to handle such updates to the *sweep* tool's settings, which is similar but not identical to function "setvar_cb". There is also a generic callback "setvar_rpc" which receives line ϕ sent from the *terminal* and passes it to the MCF subsystem. At program startup, two pseudoclosures are created (steps 1–2), one comprising a command code and a reference to "setvar_rpc"

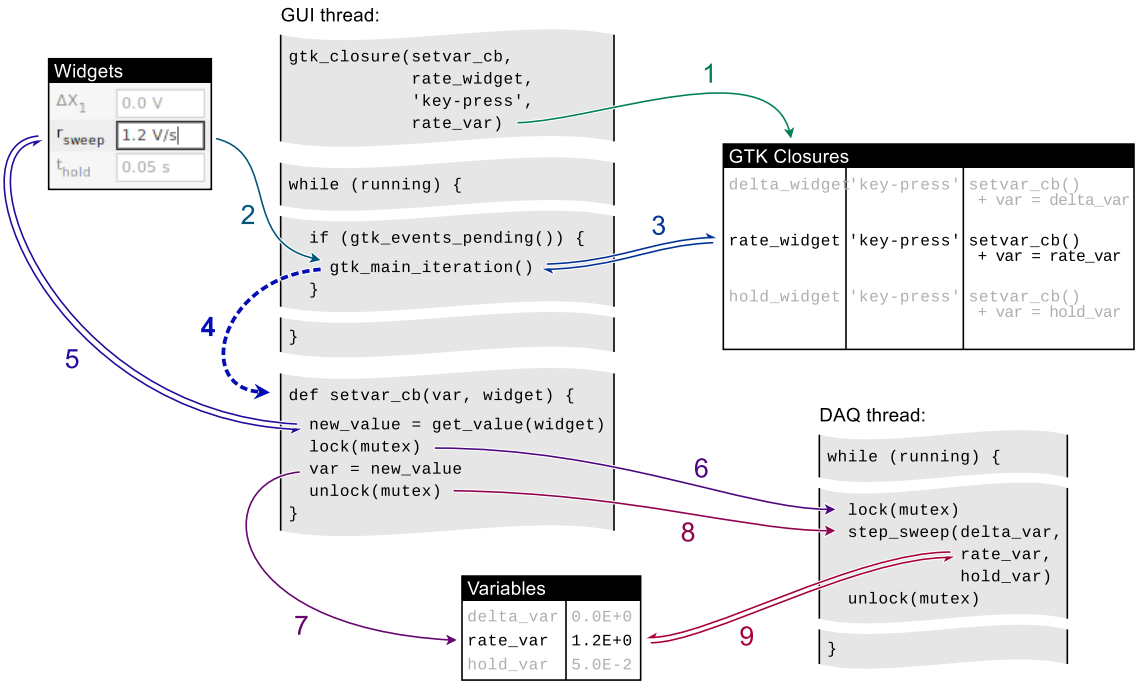


Figure 5.7: Schematic representation of the registration and prototypical calling sequence of a *GTK+* callback function. The sequence of data transfers, indicated by thin solid lines, and function calls, indicated be thick dashed lines, is overlaid on the relevant sections of *Mezurit 2* (expressed as pseudocode for clarity). Partial data structures are represented as tables.

(with no additional captured data, in this case) and another comprising the MCF key and references to "setvar_mcf", the variable, and the widget. When the user (or a script) executes the terminal command (step 3), it is received by the DAQ thread (not shown) and queued for the GUI thread, which handles the command by invoking the matching RPC pseudoclosure (steps 4–5). Line ϕ is then parsed and the resulting MCF key is matched to the pseudoclosure (step 6), which is invoked (step 7), safely updating the variable (steps 8–10). The DAQ thread will now sweep at the new rate (step 11). Finally, the appropriate widget is updated to reflect the change (step 12).

Pseudoclosures can be thought of as the way disparate parts of *Mezurit 2* are connected together, defining its large-scale structure. The collection of settings, *GTK+* widgets, MCF nodes, RPC commands, and associated callback functions for each tool and pseudotool makes up the bulk of *Mezurit 2's* data and programming. Implementing a new feature or setting simply requires adding one of each object to the source code and creating the appropriate pseudoclosures to insert the logic into the acquisition and interface loops.

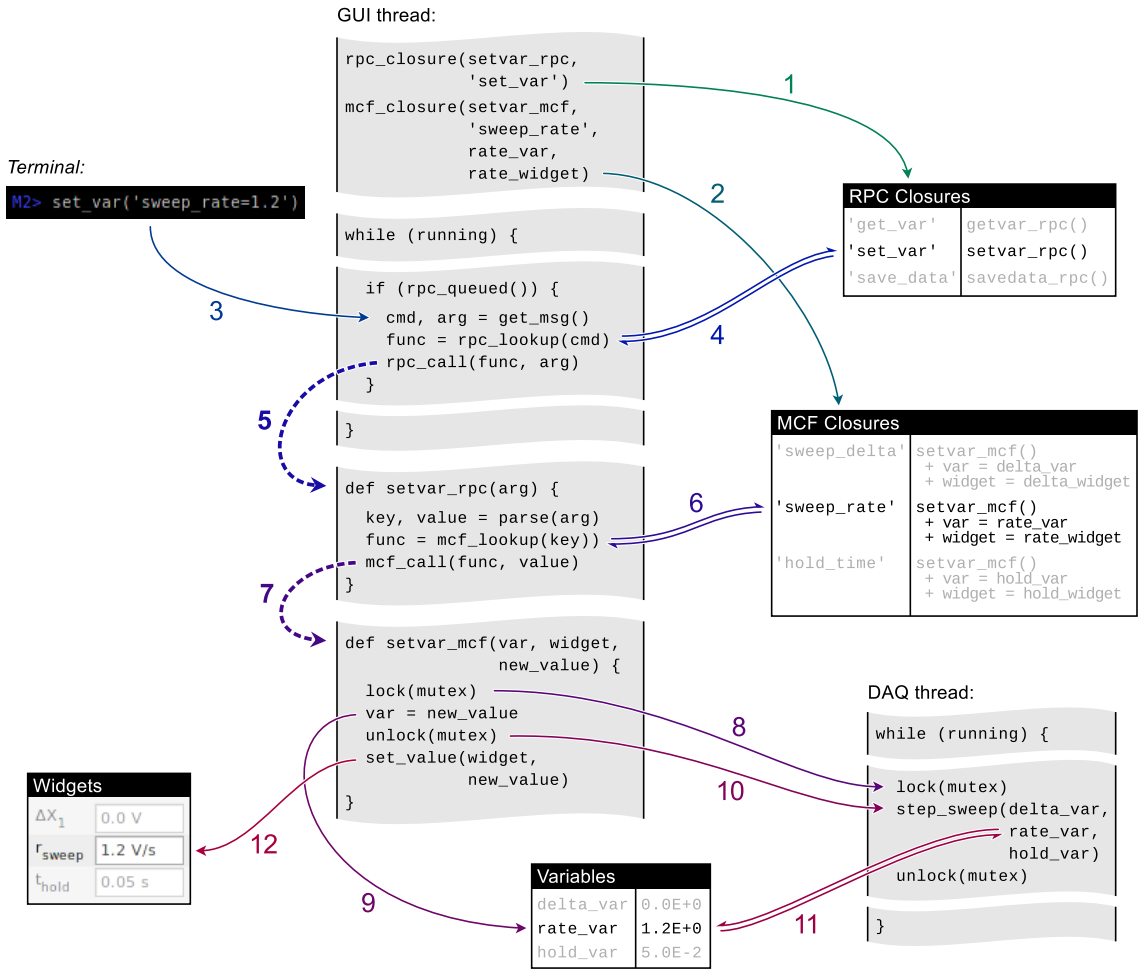


Figure 5.8: Schematic representation of the registration and prototypical calling sequence of a nested pair (RPC→MCF) of functions. The sequence of data transfers, indicated by thin solid lines, and function calls, indicated be thick dashed lines, is overlaid on the relevant sections of *Mezurit 2* (expressed as pseudocode for clarity). Partial data structures are represented as tables.

5.6 Future work

At present *Mezurit 2* meets the requirements of many common device characterization tasks. Once the (limited) selection channel of functions are memorized, the user can quickly and conveniently define virtual channels. It can sweep multiple outputs in a carefully controlled fashion and record a system's response with fine time resolution. A comprehensive scripting interface is provided for measurement automation, which is made easier to learn by embedding a command-line interface directly in the GUI.

The application may be extended in the future, especially if the number of users continues to

grow.[23] While it is hoped that the availability of the source code and its modular design will encourage others to contribute to its development, there are other architectural changes which could open up the process to non-C programmers. One possibility would be to enable pure Python-based (rather than *terminal*-based) extensions with the same GUI integration as the built-in tools. Taking this idea even further, the tools and pseudotools could be reimplemented in Python where performance considerations allow. Thus, the lower-level C code would become a library to support user-generated virtual instruments. Apart from these hypotheticals, it is hoped that more users will find that *Mezurit 2* helps them do science and perhaps share their scripts and MCF files with others.

[23] As of April 2012 *Mezurit 2* is being used in 3–4 research labs and evaluated in two more.

Chapter 6

Conclusions

The electronic and mechanical properties of graphene have been studied intensely over the last ∼8 years by researchers in many fields, from chemistry to mechanical engineering to, of course, physics. The myriad results of these studies lend support to the idea that graphene has many applications both inside and outside science. Apart from graphene's intrinsic scientific value as a two-dimensional electronic material with an unusual band structure, one might describe it as a *platform* for nanoscience and nanotechnology in general. Two examples of the latter were presented in this work—one focusing on the edge of graphene as an intrinsically nanoscale object and interface to atomic-scale carbon structures and another demonstrating that oxidized graphene is useful as a layered insulator.

Chapter 2 described graphene device fabrication techniques, which have themselves evolved considerably in recent years. While traditional graphene deposition methods were used extensively in this work, future devices will surely be mass produced from wafer-scale sheets of graphene, bringing additional experiments into the realm of possibility. Most, if not all, practitioners of the "Scotch tape" method will welcome that development.

The technique of electrical breakdown was applied to two-terminal graphene devices, as reported in chapter 3, and found to reliably create nanometer-scale gaps, or junctions. These junctions showed switching behavior when subjected to voltage pulses such that they can be programmed to be OFF, i.e., in a low-conductance state, or ON, i.e., in a high-conductance state. The devices have been operated for hundreds of thousands of switching cycles without degradation and the conductance state persists for over 24 hours. A model of electric field-driven motion of atomic chains of carbon was proposed to as a possible switching mechanism. According to this model, in the ON

state, electrons flow through carbon-atom wires bridging the nanogap. Thus, the edge of graphene serves as a connection point for a molecular electronic device.

Chapter 4 reported on the fabrication and characterization of top-gated graphene field-effect transistors made with graphite oxide as the gate dielectric. Working with graphite oxide presented a challenge to the standard processing recipes due to its temperature sensitivity. A low-temperature electron-beam lithography process was developed which maintained the integrity of the graphite oxide as measured by color contrast and transport measurements. Measurements of the channel conductance as a function of both the top-gate and doped silicon back-gate showed that graphite oxide's dielectric constant, κ, is about 4.3. Its breakdown electric field was found to be comparable to SiO_2. This work demonstrates the utility of graphite oxide as a general-purpose layered insulator compatible with graphene and probably other materials.

The final chapter described a comprehensive software application—*Mezurit 2*—which supported the measurements described above. It implements virtual instruments covering many basic electronics experiments and has a comprehensive Python-based scripting interface for customization and automation. The virtual instruments' operational models and the application's overall design were detailed.

These results will hopefully enable further progress toward the realization of graphene as a general-purpose platform for nanoscale electronics (complete with data acquisition software).

Appendix A

Conversion of lithography patterns using *npgsfixer*

A.1 Introduction

The *Nanometer Pattern Generation System* (*NPGS*) [35] is used for direct-write lithography using scanning electron or focused ion beam microscopes. In a typical usage scenario, *NPGS* is used to read one or more patterns created in *DesignCAD* into memory. A "runfile" is then generated based on their contents combined with user-supplied information about location, alignment, and dosage. Finally, *NPGS* "writes" the pattern to the sample by rapidly modulating beam deflection over high-speed X-Y inputs.

Though *DesignCAD* is integrated with *NPGS* and is itself a powerful computer-aided design (CAD) package, some prefer a work flow based on *AutoCAD* or compatible programs, such as *QCAD* [120], which read and write in the widely used Drawing Exchange Format (DXF). While *DesignCAD* is capable of importing DXF files, the resulting patterns are not directly usable by *NPGS* because *DesignCAD* has no way of knowing which shapes are physical features and which are ancillary markings, such as those used for registration or alignment.

npgsfixer is a Perl script which converts a generic *DesignCAD* file into one which is compatible with *NPGS*. Color is used to encode role information about each shape, which is then translated into the embedded flags *NPGS* expects. The script is also capable of fracturing large patterns into subfields based on user-defined bounding boxes and generating a list of X-Y stage movements for stitching the subfields back together during writing.[1]

[1] *NPGS* is capable of automating this process, but using *npgsfixer* offers more control.

A.2 Operation

Usage instructions can be printed using the '--help' option:

```
>> npgsfixer --help

Usage: npgsfixer [input file] [output file] [arguments]

  Arguments:  --fieldbase [file]  Specify base filename of subfields,
                                  if applicable. Default: "field"
              --runfile   [file]  Specify filename of runfile snippet,
                                  if applicable. Default: "runfile.txt"
```

The input file should be a *DesignCAD* file containing color-coded shapes that comprise the pattern to be written. Physical shapes must be closed "polylines" of one of six (to accommodate dose ranges) predefined colors.[2] *npgsfixer* runs in one of two modes, depending on the contents of the pattern:

Normal mode Sets the appropriate flags on the shapes in the pattern according to color-based encoding. Unmatched shapes are passed through unchanged.

Fracture mode Automatically activated when at least one layer containing a yellow bounding box is detected, the pattern is fractured into separate files, one per boxed layer. Coordinates are adjusted to place the origin of each subfield in the center of its bounding box and a list of stage movements for stitching is generated.

The same translation of color-coding to embedded flags occurs in both normal mode and fracture mode. In the latter case, shapes in unboxed layers are simply written to the main output file.

Fracturing is often used to take advantage of the higher resolution offered by writing at a larger magnification without changing the overall dimensions of the pattern. The quality of the stitching depends on the accuracy of the motorized stage fitted on the microscope and the center-to-center distance D between subfields. The manual control over the bounding boxes offered by *npgsfixer* can be exploited to reduce $\langle D \rangle$ or ensure that critical areas fall within the same subfield. In addition, overlapping subfields can be used to separate interspersed fine and course features into separate writing steps.

[2] Light blue, medium blue, dark blue, light green, medium green, and dark green. Their exact definitions can be found in the script itself, where additional options may easily be added.

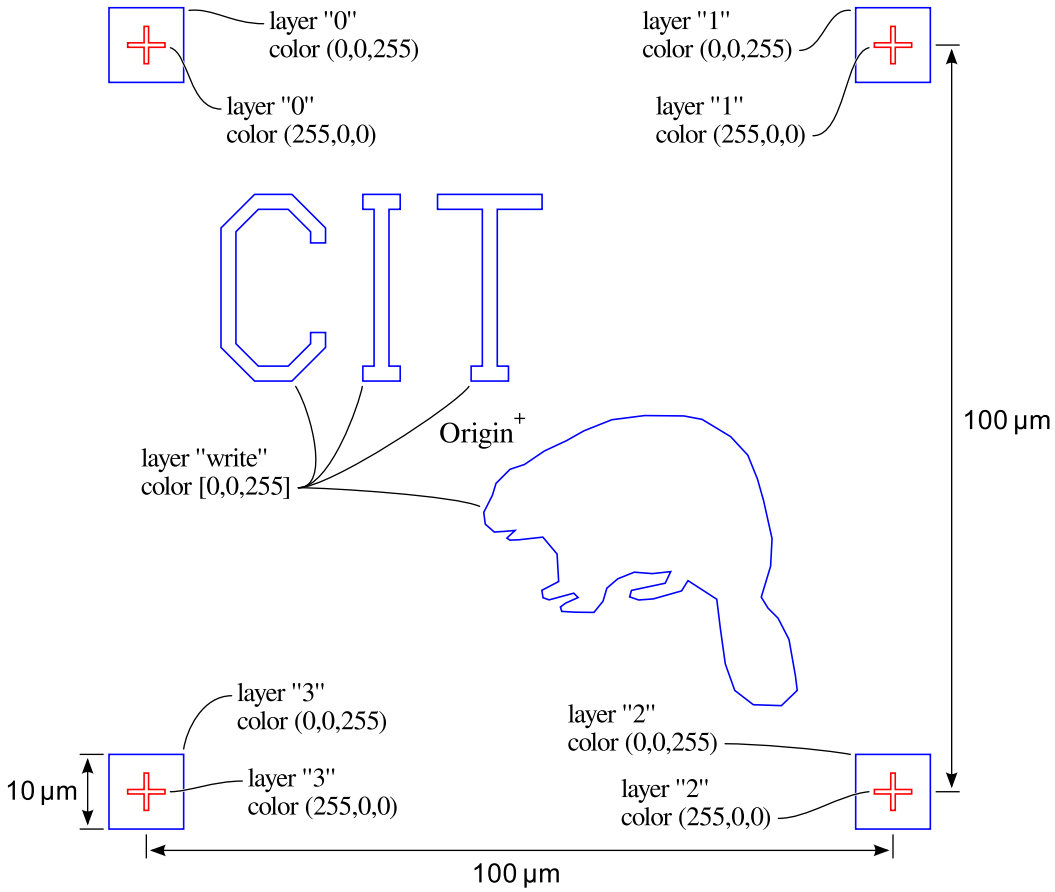


Figure A.1: Example lithography pattern defining four alignment windows (layers “0” through “3”) and one writing step (layer “write”)

A.3 Examples

A.3.1 Normal mode

Normal mode operation is illustrated using the test pattern shown in Figure A.1, which defines four alignment windows with target markers and several features to be written. Given that no bounding boxes are to be found, the blue shapes are flagged as physical shapes and the red marks are passed through unchanged:

```
>> npgsfixer example_normal_in.DC2 example_normal_out.DC2
...
SECOND PASS:
Processing layer 1 "0":
  Fixed filled polyline. Vertices: 4 Color: (0,0,255)
  Skipped nonfilled polyline. Vertices: 12 Color: (255,0,0)
...
Processing layer 5 "write":
  Fixed filled polyline. Vertices: 12 Color: (0,0,255)
```

```
  Fixed filled polyline. Vertices: 12 Color: (0,0,255)
  Fixed filled polyline. Vertices: 20 Color: (0,0,255)
  Fixed filled polyline. Vertices: 63 Color: (0,0,255)
...
```

The output file (example_normal_out.DC2) can then be loaded into *NPGS* via a runfile specifying one manual alignment[3] step, using layers 1–4, and one writing step, using layer 5. Screen captures of the alignment and writing processes are shown in Figure A.2.

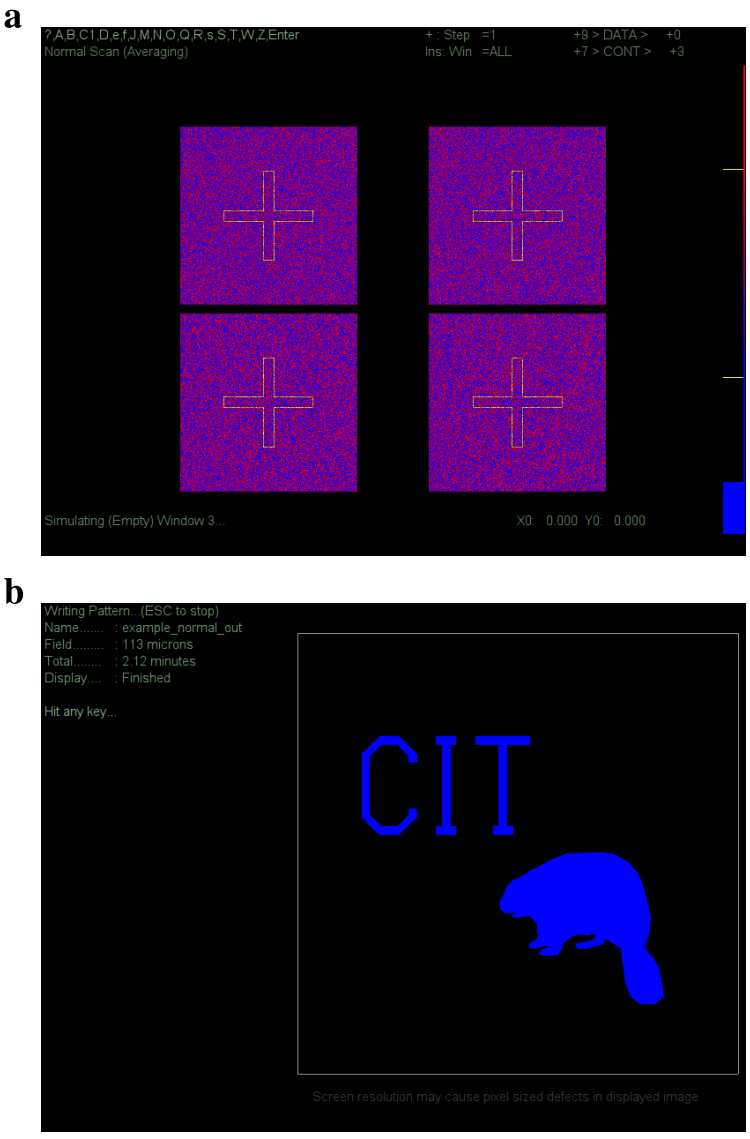


Figure A.2: *NPGS* screen captures while processing an example pattern. (a) Alignment step with four windows. (b) Writing step

[3] Alignment windows are a special case of physical shapes which define a subsection of the field to image during the alignment process. They are typically not filled during writing.

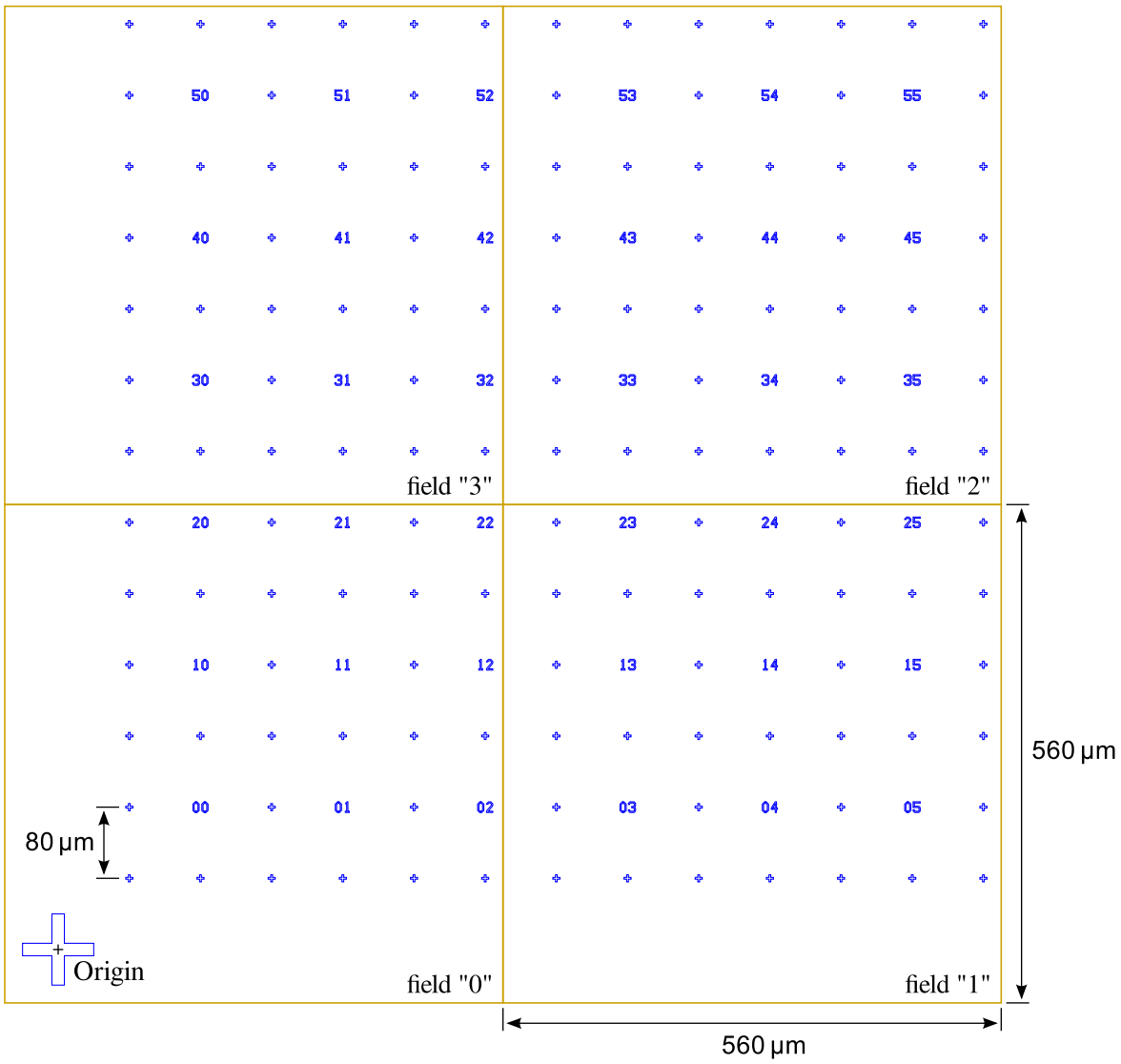


Figure A.3: Example lithography pattern defining a grid with four subfields

A.3.2 Fracture mode

Fracture mode operation is illustrated using the test pattern shown in Figure A.3, which contains a sparse grid of alignment marks. *npgsfixer* will detect the yellow bounding boxes and split the pattern into four subfields:

```
>> npgsfixer example_fracture_in.DC2 example_fracture_out.DC2 \
--fieldbase exfield --runfile exrun.txt
...
FIRST PASS:
Scanning layer 1 "0":
  Found bounding box. Vertices: 4, Center: (220.0, 220.0)
```

```
Scanning layer 2 "1":
  Found bounding box. Vertices: 4, Center: (780.0, 220.0)
Scanning layer 3 "2":
  Found bounding box. Vertices: 4, Center: (780.0, 780.0)
Scanning layer 4 "3":
  Found bounding box. Vertices: 4, Center: (220.0, 780.0)
SECOND PASS:
Processing layer 1 "0":
  Fixed filled polyline. Vertices: 12 Color: (0,0,255)
  Fixed filled polyline. Vertices: 12 Color: (0,0,255)
  ...
  Deleted bounding box.
...
```

The subfields are written as separate files (exfield-0.DC2, exfield-1.DC2, exfield-2.DC2, and exfield-3.DC2). Given that all four layers include bounding boxes, the main output file contains no shapes and can be discarded. In addition, a list of X-Y stage movements is saved as a runfile snippet (exrun.txt) describing the stitching scheme for the overall pattern, shown in Figure A.4. A full-fledged runfile can then be created from the snippet which writes field "0" centered at location r_0, field "1" at r_1, field "2" at r_2, and field "3" at r_3. Screen captures of the writing processes are shown in Figure A.5.

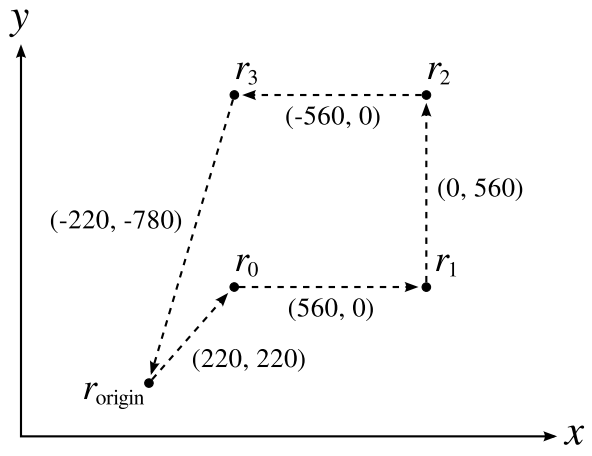


Figure A.4: X-Y stage movements required to stitch a fractured pattern. The stage moves from position r_{origin} to r_0 before the first writing step, then to r_1 for the second step, and so on. After the final step, the stage returns to r_{origin}.

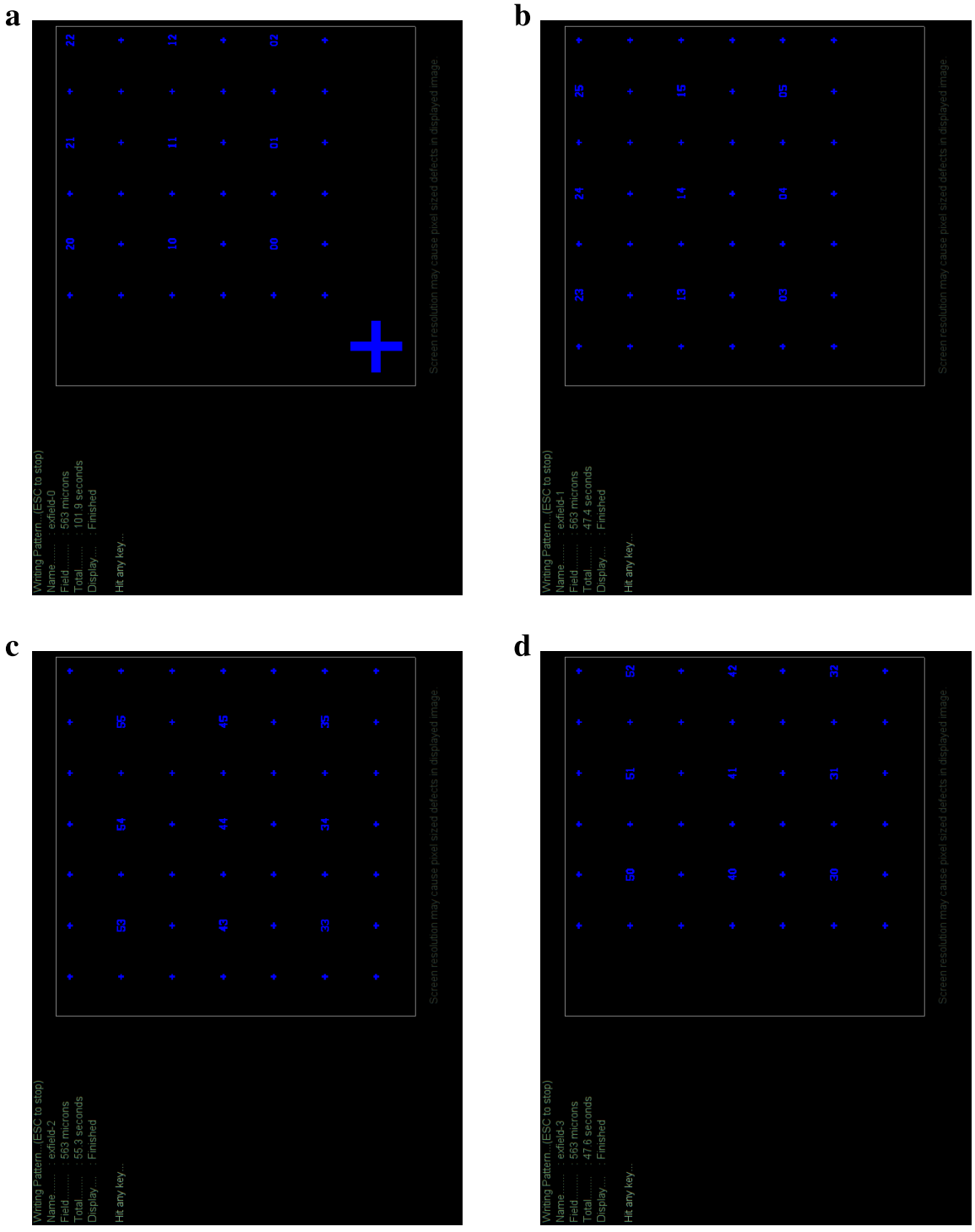


Figure A.5: *NPGS* screen captures (rotated 90°) while writing a fractured pattern's four subfields: (a) field "0", (b) field "1", (c) field "2", and (d) field "3"

A.4 Code listing

```
#!/usr/bin/perl

#  Copyright (C) 2012 Brian Standley
#
#  This program is free software: you can redistribute it and/or modify it
#  under the terms of the GNU General Public License as published by the
#  Free Software Foundation, either version 3 of the License, or (at your
#  option) any later version.
#
#  This program is distributed in the hope that it will be useful, but
#  WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
#  Public License for more details.
#
#  You should have received a copy of the GNU General Public License along
#  with this program. If not, see <http://www.gnu.org/licenses/>.

# npgsfixer v1.1t

$infilename   = shift @ARGV;
$mainfilename = shift @ARGV;
$fieldname    = "field";
$runfilename  = "runfile.txt";

while (@ARGV > 0)
{
   $arg = shift @ARGV;
   if    ($arg =~ "--fieldbase" && @ARGV > 0) { $fieldname   = shift @ARGV; }
   elsif ($arg =~ "--runfile"   && @ARGV > 0) { $runfilename = shift @ARGV; }
}

if ($infilename eq "--help" || $infilename eq "-h")
{
   print "\nUsage: npgsfixer [input file] [output file] [arguments]\n\n";
   print "  Arguments:  --fieldbase [file]  Specifiy base filename of" .
         " subfields,\n                                  " .
         "  if applicable. Default: \"field\"\n";
   print "              --runfile   [file]  Specifiy filename of runfile" .
         " snippet,\n                                  " .
         "  if applicable. Default: \"runfile.txt\"\n\n";
   exit 0;
}

print "\nConverting input file to UNIX format:\n";
system "dos2unix --verbose --d2u $infilename";

sub die2
{
   printf "\nConverting input file back to DOS format:\n";
   system "dos2unix --verbose --u2d $infilename";
   $msg = shift @_;
   die $msg;
}

!($infilename =~ $mainfilename) || die2("Input and output files must be" .
                                        " different.\n");
```

```
open(INFILE,   "< $infilename") || die2("Cannot open file: $infilename.\n");

print "\nFIRST PASS:\n";

$header = "";
while (my $line = <INFILE>)
{
   if ($line =~ /23 (.+) 0 0 0 0/)
   {
      for (my $L = 0; $L < $1; $L++)
      {
         $chop[$L] = 0;
         $name[$L] = <INFILE>;
         chomp($name[$L]);
      }
      last;
   }
   $header = $header . $line;
}

@YELLOW      = (255, 255,   0);  # QCAD: (255, 255,   0)
@BLUE        = (  0,   0, 255);  # QCAD: (  0,   0, 255)
@GREEN       = (  0, 255,   0);  # QCAD: (  0, 255,   0)
@LIGHT_BLUE  = (128, 191, 255);  # QCAD: (175, 175, 255)
@LIGHT_GREEN = (191, 255, 128);  # QCAD: (175, 255, 175)
@DARK_BLUE   = (  0,   0, 166);  # QCAD: (  0,   0, 150)
@DARK_GREEN  = (  0, 166,   0);  # QCAD: (  0, 150,   0)

sub MatchColor
{
   my ($c1, $c2) = @_;
   return ($$c1[0] == $$c2[0] && $$c1[1] == $$c2[1] && $$c1[2] == $$c2[2]);
}

$N_field = 0;
while (my $line = <INFILE>)
{
   if ($line =~ /21 (.+) 0 0 0 0/)
   {
      $L = $1;
      print "Scanning layer $L \"$name[$L]\":\n";
   }
   elsif ($line =~ /1 (.+) [68] .+ 0 (.+) (.+) (.+) 10 1/)
   {
      my @color = ($2, $3, $4);
      if (MatchColor(\@color, \@YELLOW))
      {
         my ($x0, $x1, $y0, $y1);
         for (my $i = 0; $i < $1; $i++)
         {
            my ($x, $y, $z) = split(' ', <INFILE>);

            if ($i == 0 || $x < $x0) { $x0 = $x; }
            if ($i == 0 || $x > $x1) { $x1 = $x; }
            if ($i == 0 || $y < $y0) { $y0 = $y; }
            if ($i == 0 || $y > $y1) { $y1 = $y; }
         }
```

```
            $N_field++;
            $chop[$L] = 1;
            $xc[$L] = ($x1 + $x0) / 2.0;
            $yc[$L] = ($y1 + $y0) / 2.0;

            printf "  Found bounding box. Vertices: %d,", $1 - 1;
            printf " Center: (%1.1f, %1.1f)\n", $xc[$L] / 8.0, $yc[$L] / 8.0;
         }
      }
   }

   if ($N_field > 0)
   {
      open(RUNFILE, "> $runfilename") || die2("Cannot open file: $runfilename.\n");

      my ($xs, $yx) = (0.0, 0.0);
      for (my $L = 0; $L < @name; $L++)
      {
         if ($chop[$L])
         {
            printf RUNFILE "$fieldname-$name[$L]\n1\n%d,%d\n",
                           ($xc[$L] - $xs) / 8.0, ($yc[$L] - $ys) / 8.0;
            $xs = $xc[$L];
            $ys = $yc[$L];
         }
      }
      printf RUNFILE "MoveOnly M\n%d,%d\n", $xs / -8.0, $ys / -8.0;

      close(RUNFILE);
      system "dos2unix --u2d $runfilename";
   }

   open(MAINFILE, "> $mainfilename") || die2("Cannot open file: $mainfilename.\n");

   print  MAINFILE $header;
   printf MAINFILE "23 %d 0 0 0 0\n", @name - $N_field;
   for (my $L = 0; $L < @name; $L++)
   {
      if ($chop[$L] == 0) { print MAINFILE "$name[$L]\n"; }
   }

   printf "SECOND PASS:\n";

   seek(INFILE, 0, 0);

   $vertex_mode = 0;
   $chopped_so_far = 0;
   while (my $line = <INFILE>)
   {
      if ($line =~ /21 (.+) 0 0 0 0/)
      {
         $L = $1;
         print "Processing layer $L \"$name[$L]\":\n";

         if ($chop[$L])
         {
            if ($chopped_so_far > 0)
            {
```

```
            close(CHOPFILE);
            system "dos2unix --u2d $chop_filename";
         }

         $chop_filename = "$fieldname-$name[$L].DC2\n";
         open(CHOPFILE, "> $chop_filename") || die2("Cannot open file:" .
                                                   " $chop_filename.\n");

         print CHOPFILE $header;
         print CHOPFILE "23 2 0 0 0 0\n";
         print CHOPFILE "$name[0]\n";
         print CHOPFILE "$name[$L]\n";
         print CHOPFILE "21 1 0 0 0 0\n";

         $chopped_so_far++;
      }
      else { printf MAINFILE "21 %d 0 0 0 0\n", $L - $chopped_so_far; }

      $vertex_mode = 0;
   }
   elsif ($line =~ /1 (.+) [68] .+ 0 (.+) (.+) (.+) 10 1/)
   {
      my @color = ($2, $3, $4);
      if (MatchColor(\@color, \@YELLOW))
      {
         $vertex_mode = 0;
         print "  Deleted bounding box.\n";
      }
      elsif (MatchColor(\@color, \@BLUE)        ||
             MatchColor(\@color, \@GREEN)       ||
             MatchColor(\@color, \@LIGHT_BLUE)  ||
             MatchColor(\@color, \@LIGHT_GREEN) ||
             MatchColor(\@color, \@DARK_BLUE)   ||
             MatchColor(\@color, \@DARK_GREEN))
      {
         my $fixed = "1 $1 16 0 1 12 0 0 0 $2 $3 $4 10 1\n";
         if ($chop[$L]) { printf CHOPFILE $fixed; }
         else           { printf MAINFILE $fixed; }

         $vertex_mode = 1;
         printf "  Fixed filled polyline. Vertices: %d Color: ($2,$3,$4)\n",
                $1 - 1;
      }
      else
      {
         if ($chop[$L]) { printf CHOPFILE $line; }
         else           { printf MAINFILE $line; }

         $vertex_mode = 1;
         printf "  Skipped nonfilled polyline. Vertices: %d Color: ($2,$3,$4)\n",
                $1 - 1;
      }
   }
   elsif ($line =~ /1 (.+) 16 0 1 12 0 0 0 (.+) (.+) (.+) 10 1/)
   {
      if ($chop[$L]) { printf CHOPFILE $line; }
      else           { printf MAINFILE $line; }
```

```
      $vertex_mode = 1;
      printf "  Skipped filled polyline. Vertices: %d Color: ($2,$3,$4)\n",
             $1 - 1;
   }
   elsif ($vertex_mode)
   {
      if ($chop[$L])
      {
         my ($x, $y, $z) = split(' ', $line);
         printf CHOPFILE "%d %d 0\n", $x - $xc[$L], $y - $yc[$L];
      }
      else { print MAINFILE $line; }
   }
}

close(INFILE);
close(MAINFILE);
system "dos2unix --u2d $mainfilename";

if ($chopped_so_far > 0)
{
   close(CHOPFILE);
   system "dos2unix --u2d $chop_filename";
}

printf "\nConverting input file back to DOS format:\n";
system "dos2unix --verbose --u2d $infilename";

exit 0;
```

Appendix B

80 V bipolar gate voltage amplifier

B.1 Introduction

Nanoscale electronics experiments are often conducted using devices built on top of degenerately-doped silicon wafers covered by a thin layer of oxide. In many graphene-based devices, the bulk Si is used as an electrostatic gate by applying a voltage via a backside contact. Achieving adequate electric field, however, can be a challenge due to SiO_2's modest permittivity ($3.9\,\epsilon_0$) and constraints on its thickness imposed by the mechanically exfoliation process [1, 27].[1] For example, the device featured in section 4.3.3 required 71 V on the back-gate to effect the same doping as applying just 4 V to the top-gate. Other dual-gated devices with similar oxide thickness have been measured with $V_{bg} > 150\,\text{V}$ [121].

The digital-to-analog converters (DACs) provided by the computer-based data acquisition cards used in many electrical measurement systems are typically limited to $\pm 10\,\text{V}$. Instruments such as the Keithley 2400-series SourceMeters meet the voltage requirements, but can be difficult to integrate into an analog system. Given these considerations, a custom bipolar gate voltage amplifier (BGVA) was designed and constructed to convert a DAC-sized signal into a graphene-scale back-gate voltage. It is capable of smoothly sweeping from $-80\,\text{V}$ to $80\,\text{V}$ with 1% accuracy and has a nominal DC voltage gain A_V of 10 V/V such that the required input voltage is limited to $\pm 8\,\text{V}$. A photo of its front panel is shown in Figure B.1.

[1] Blake et al. [27] report that graphene is most visible on 80 nm and 280 nm SiO_2 layers, while Novoselov et al. [1] (Supporting Online Material) chose a thicker, 300 nm layer to avoid damage during fabrication.

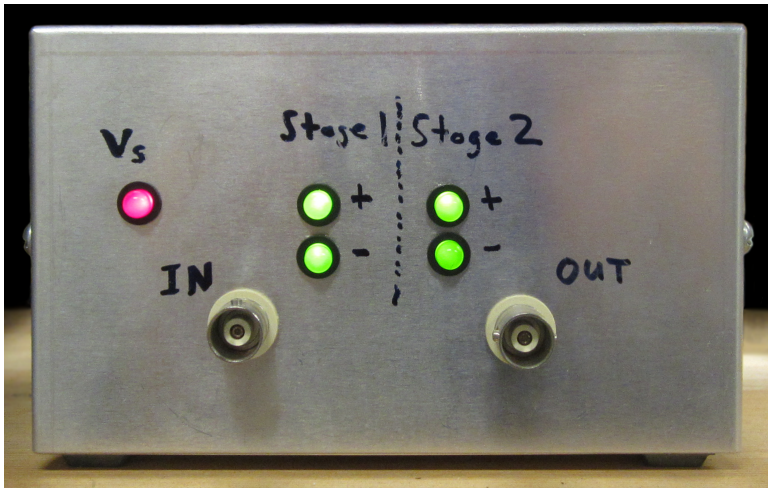


Figure B.1: Photo of bipolar gate voltage amplifier front panel, including input and output BNCs and LEDs indicating the status of the power supply and stage rail voltages. The back panel (not shown) includes "banana" jacks for the power supply and a fuse holder.

B.2 Specifications and performance

The BGVA requires a single 15 V external power supply, which is electrically isolated from the input and output connections to eliminate a potential ground loop. When paired with the (included) external filter, the amplifier produces no more than $\approx$1 $\mathrm{mV_{rms}}$ output noise while maintaining more than 1 kHz of bandwidth. Although the BGVA is a prototype, it offers standard safety features such as limited short circuit protection and a fused supply connection. In the event that the power supply polarity is reversed, included protection diodes should prevent damage to the amplifier. Complete specifications are given in Table B.1.

The BGVA's performance exceeds the requirements of simple DC gate sweep measurements. The voltage gain A_V for a sinusoidal input signal V_i is plotted as a function of frequency f in Figure B.2a. The amplified signal V_o (measured the output of the external filter) falls to 1% of its DC value at 1.2 kHz and is reduced by 3 dB at 12.3 kHz. The step response rises to 99% of its final value in 156 µs with no ringing or overshoot, as shown in Figure B.2b. (Without the external filter, rise time is 58 µs with a 0.3% overshoot.)

Table B.1: Amplifier specifications with external filter installed

	Min.	Nominal	Max.	Unit
GAIN				
Voltage gain, DC	9.9	10.0	10.1	V/V
Bandwidth (1%)		1.2[a]		kHz
Bandwidth (−3 dB)		12.3[a]		kHz
INPUT				
Positive input voltage		8.0	8.3	V
Negative input voltage	−8.3	−8.0		V
Input resistance	-	1	-	MΩ
OUTPUT				
Positive output voltage		80	83	V
Negative output voltage	−83	−80		V
Output resistance	-	17.4[b]	-	kΩ
POWER SUPPLY				
Power supply voltage	14.5	15	15.5	V
Power supply current	130	160	180	mA
Power supply fuse	-	0.5	-	A

[a] The bandwidth is slightly greater without the external filter.
[b] The output resistance is 8 kΩ without the external filter.

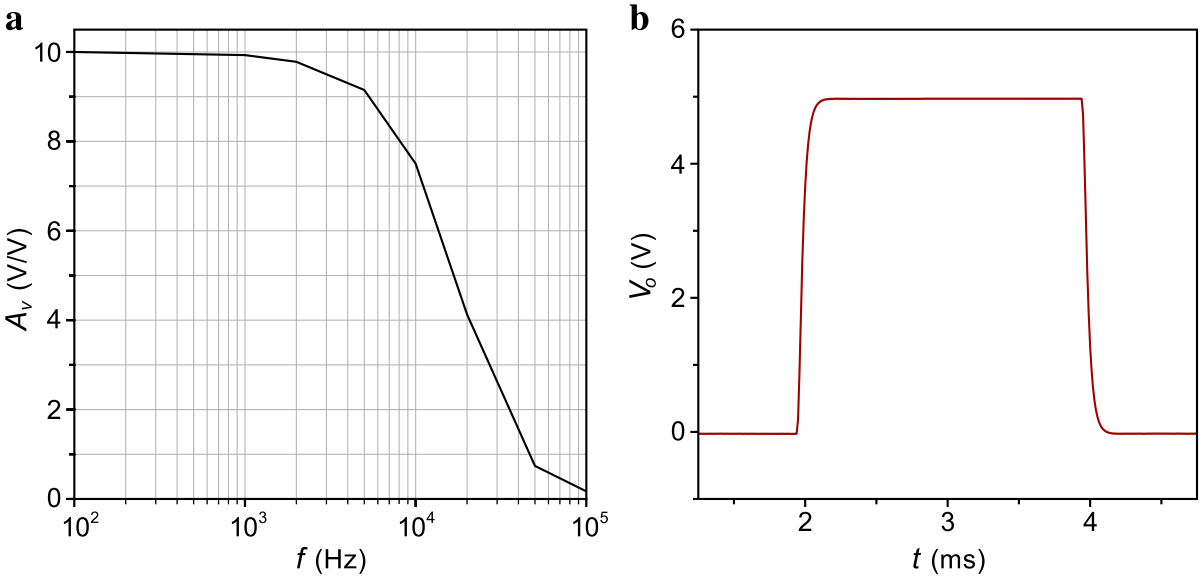


Figure B.2: Amplifier performance. (a) Voltage gain vs. frequency for a sinusoidal input signal, with external filter installed. (b) Output voltage for a 0.5 V square input signal, demonstrating over-damped response

B.3 Operating instructions

Table B.2: Absolute maximum ratings for bipolar gate voltage amplifier

Power supply voltage	.....	17 V
Input voltage, unpowered	.....	$\pm$3 V
Input voltage, powered	.....	$\pm$10 V

The BGVA should be powered-up using a "soft start" procedure, as follows:

1. Switch on a DC power supply, set the output to 0 V, and the current compliance to approximately 250 mA.
2. Connect the power supply to the amplifier using "banana connector" leads.
3. Ramp the voltage to 15 V. The current compliance indicator on the power supply may flicker when passing through 4 V as the DC/DC converters come online.
4. Make the input and output connections.

The input should be disconnected or held at 0 V when the amplifier is not powered. The ratings given in Table B.2 should never be exceeded to avoid causing permanent damage to the device.

B.4 Circuit design

The BGVA comprises two noninverting, op-amp-based stages in series ($A_{V,1}$ = 5 V/V, $A_{V,2}$ = 2 V/V). This design reduces the required per-stage voltage swing to $\pm$40 V, enabling the use of widely available Texas Instruments (TI) OPA445 op-amps. Each stage is powered by three TI DCP011515DB isolated DC/DC converters connected in series to provide $\pm$45–47 V, which is then regulated down to $\pm$43 V by a pair of LM317/LM337 voltage regulators. The second stage's power supply is referenced to the output of the first stage itself, effectively level-shifting its supply rails V_{2-} and V_{2+} to match the required range. For example, given V_i = 7 V, the interstage voltage V_o will be 35 V and V_{2-} and V_{2+} will be −8 V and 78 V, respectively. Thus, the 75 V output will fall safely within the second stage voltage rails. Both stages' supply voltages along with V_o and V_x are shown as a function of V_i in Figure B.3.

Apart from the unusual supply rail shifting feature, standard techniques were employed in the BGVA design. The complete circuit schematics are shown in Figure B.4 and the components are

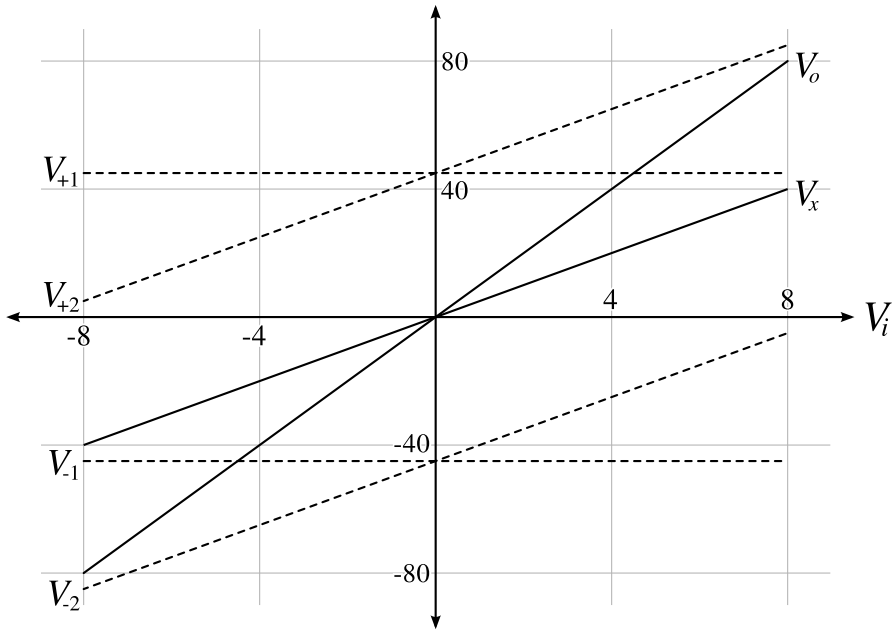


Figure B.3: Absolute supply voltages (dashed lines) for the first (V_{1-}, V_{1+}) and second (V_{2-}, V_{2+}) stages of the amplifier circuit as a function of input voltage. Interstage voltage V_x and output voltage V_o are also plotted (solid lines). The second stage's power supply is referenced to V_x.

listed in Table B.3. Power supply noise is controlled by 1 µF bypass capacitors (C11–C28) placed near the inputs and outputs of the DC/DC converters. In addition, large 4.7 µF tantalum capacitors (C1–C4) are placed near the supply terminals of the op-amps. Given that the input terminals of the DC/DC converters are isolated from their output terminals, the input power negative lead was deliberately not connected to amplifier ground to eliminate a potential ground loop in the measurement setup.

The TI DCP011515DB provides isolation by using an 800 kHz oscillator driving a integrated transformer, which is connected to a rectifier at the output. Inevitably, the transformers inductively couple to the circuit traces, generating high-frequency noise.[2] This noise was significantly reduced, however, by a 15 kHz low-pass filter enclosed in a separate aluminum box. This approach reduces the root-mean-square output noise from over 5 mV to approximately 1 mV.

[2] Batteries were considered as a low-noise alternative, but ultimately rejected in favor of a design that allowed indefinite operation.

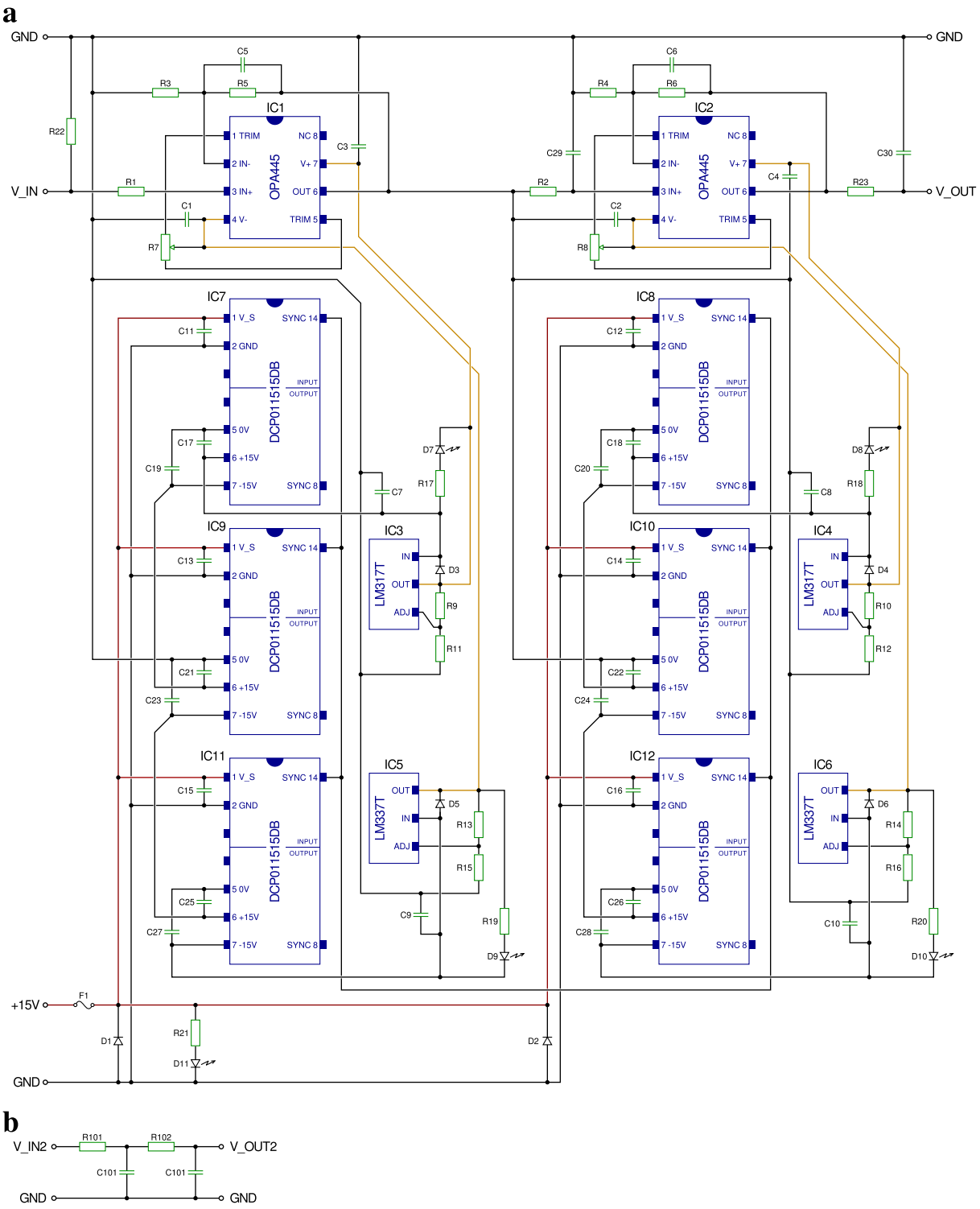


Figure B.4: Amplifier circuit schematics. (a) Main circuit. (b) External low-pass filter connected to the output of (a). The schematics are color coded for clarity: passive components are green, integrated circuits are blue, input power rails are red, and internal power rails are orange. Connection points are indicated by black dots; wire crossings lacking dots are not physically connected.

Table B.3: List of amplifier components and their values

Qty.	Type	Label	Value	Unit	Rating
2	OPA445AP	IC1,IC2	-		-
2	LM317T	IC3,IC4	-		-
2	LM337T	IC5,IC6	-		-
6	DCP011515DB	IC7-IC12	-		-
6	1N4002	D1-D6	-		1 A, 100 V
4	LED (green)	D7-D10	-		-
1	LED (red)	D11	-		-
1	fuse	F1	0.5	A	250 V
4	tantalum capacitor	C1-C4	4.7	µF	50 V
1	ceramic disc capacitor	C5	18	pF	1 kV
1	ceramic disc capacitor	C6	27	pF	1 kV
4	tantalum capacitor	C7-C10	1.0	µF	50 V
18	ceramic bypass capacitor	C11-C28	1.0	µF	50 V
1	ceramic disc capacitor	C29	47	pF	1 kV
1	ceramic disc capacitor	C30	560	pF	1 kV
1	ceramic disc capacitor	C101	688[a]	pF	1 kV
1	ceramic disc capacitor	C102	815[a]	pF	1 kV
1	resistor	R1	80.5[a]	kΩ	-
1	resistor	R2	118.6[a]	kΩ	-
1	resistor	R3	100.2[a]	kΩ	-
1	resistor	R4	244.0[a]	kΩ	-
1	resistor	R5	402[a]	kΩ	-
1	resistor	R6	243.2[a]	kΩ	-
2	trimming potentiometer	R7,R8	100	kΩ	-
1	resistor	R9	32.40[a]	kΩ	-
1	resistor	R10	32.12[a]	kΩ	-
1	resistor	R11	472[a]	kΩ	-
1	resistor	R12	465[a]	kΩ	-
1	resistor	R13	24.29[a]	kΩ	-
1	resistor	R14	24.30[a]	kΩ	-
1	resistor	R15	356.0[a]	kΩ	-
1	resistor	R16	356.2[a]	kΩ	-
4	resistor	R17-R20	2	kΩ	-
1	resistor	R21	10	kΩ	-
1	resistor	R22	1	MΩ	-
1	resistor	R23	8	kΩ	-
1	resistor	R101	5.51[a]	kΩ	-
1	resistor	R102	3.875[a]	kΩ	-

[a] This value is critical to the operation of the circuit and should be replaced within the tolerance indicated number of significant figures provided.

Appendix C

Pulse train generation using *awcgen*

C.1 Introduction

The SRS DS345 Function Generator can output arbitrary waveforms defined by a sequence of up to 16 300 points with a maximum 40 MHz timebase. The manufacturer provides a Windows program called *Arbitrary Waveform Composer* (*AWC*) [122] to generate compatible waveforms and transfer them to the instrument over a serial port or general purpose interface bus (GPIB). Unfortunately, the waveform-creation interface can be a bit cumbersome to use. *awcgen* is a Perl script which transforms a simple text file defining a piecewise-linear function into a discrete-time waveform which can then be loaded into *AWC* and transferred to the instrument.

C.2 Operation

Usage instructions can be printed using the '--help' option:

```
>> awcgen --help

Usage: awcgen [input file] [output file] [flags]

  Flags:  t  'Test mode'    Produce two-column output for easy plotting.
          m  'More points'  Attempt to use ~16000 points instead of ~1600.
          f  'Full scale'   Produce a full-size waveform when driving a
                            50 Ohm load.
```

The input file should be a list of time periods τ_i in seconds and target values V_i in volts, separated by whitespace, one line per point. *awcgen* sets the initial value to 0 V, then ramps linearly to each V_i over τ_i in succession, as illustrated in Figure C.1. Abrupt jumps can be made by setting $\tau_i = 0$.

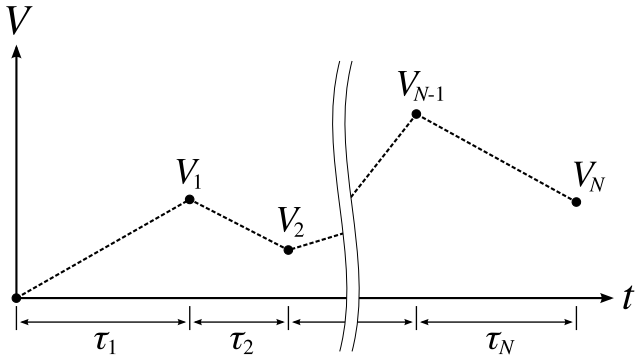


Figure C.1: Scheme used by *awcgen* to describe an arbitrary piecewise-linear waveform

In preparing the waveform, *awcgen* attempts to choose a sample rate f_{samp} that meets three goals, listed below in order of importance:

1. Satisfies: f_{samp} = (40 MHz)/N where $1 \leqslant N \leqslant 2^{34} - 1$
2. Produces about 1600 points (to reduce transfer time).
3. Is a round number.

Complex waveforms, particularly those containing both large and small τ_i, may require more points to be faithfully reproduced. Supplying the 'm' flag will maximize the number points available at the cost of a much longer transfer time. The output will be an *AWC*-compatible text file containing several lines of header information followed by a list of voltages V_j at times $t_j = \{0, t_{samp}, 2t_{samp}, ...\}$ where $t_{samp} = 1/f_{samp}$.

C.3 Example

Program operation is illustrated using an example input file (example_pwl.txt) as follows:

```
>> cat example_pwl.txt

100e-3 0
0      0.8
100e-3 0.8
0      0
200e-3 0
0      -0.05
50e-3  0.05
0      0
100e-3 0
```

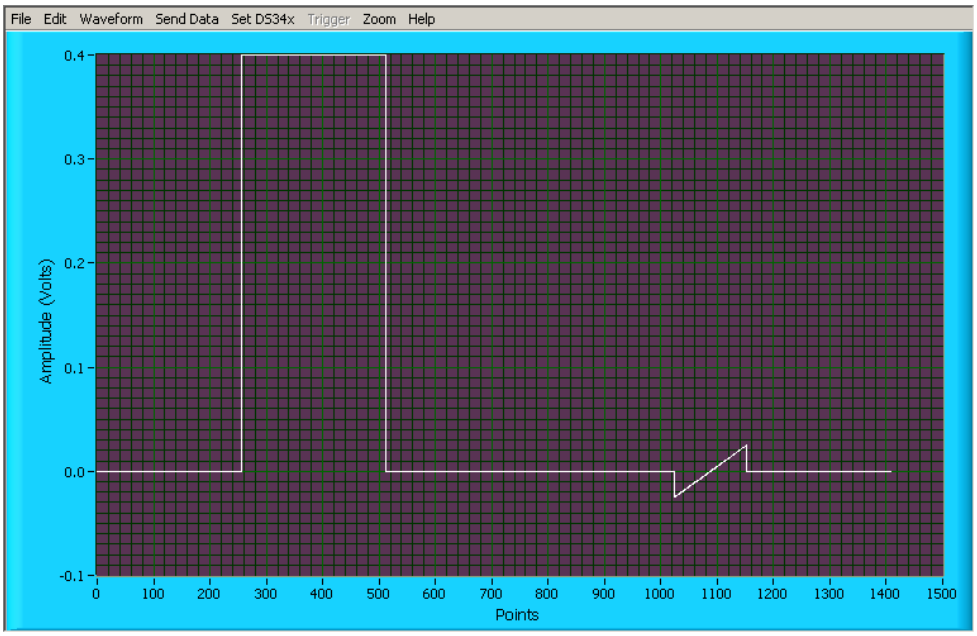


Figure C.2: *awcgen* output: Image showing an example *awcgen*-generated waveform loaded into *AWC*

```
>> awcgen example_pwl.txt example_awc.txt

             Request       Actual
             ------------  ------------
Divider:     13750.000     15625
Frequency:   2.909e+03     2.560e+03
N_points:    1600          1409

Total time:  5.500e-01
Output:      AWC-compatible
```

The output file (example_awc.txt) can then be opened in *AWC*, and would look similar to Figure C.2. The waveform time scale may be verified by generating a continuous function $V(t)$ instead of V_j using the 't' flag:

```
>> awcgen example_pwl.txt example_test.txt t
...
Output:      Plot-compatible
```

This output file (example_test.txt) can then be easily loaded by plotting software, as shown in Figure C.3.

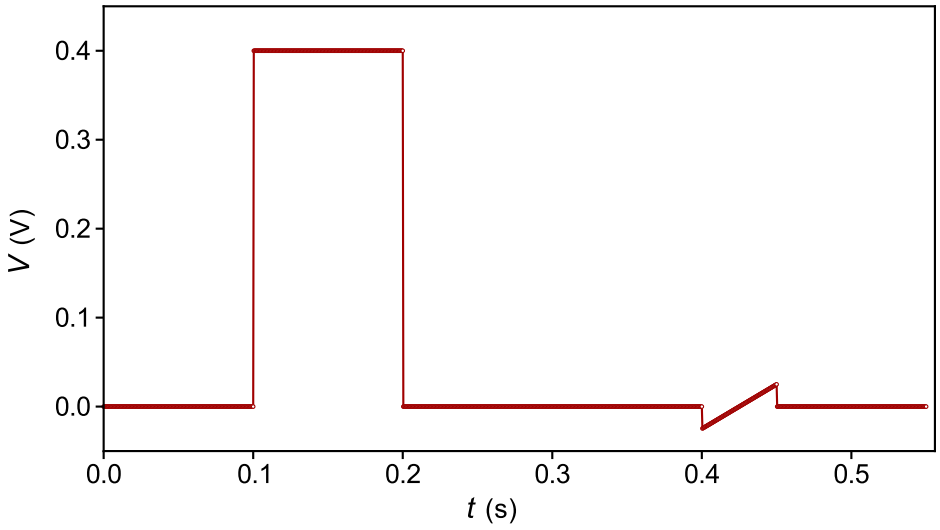


Figure C.3: *awcgen* output: Plot of an example *awcgen*-generated waveform

C.4 Load impedance

Note that the amplitude of the example waveform appears to be half of that specified in the input file. The reason is related to assumptions about load impedance: The instrument provides a 50 Ω output, and expects a 50 Ω load, so it scales the internal source voltage V_s to produce the expected output voltage in that situation, illustrated in Figure C.4a. In contrast, *awcgen* was written with a high-impedance load in mind (i.e., Figure C.4b), so by default it divides all voltages by two to compensate for the instrument's assumption. To drive a 50 Ω load, simply supply the 'f' flag as mentioned in the usage instructions.

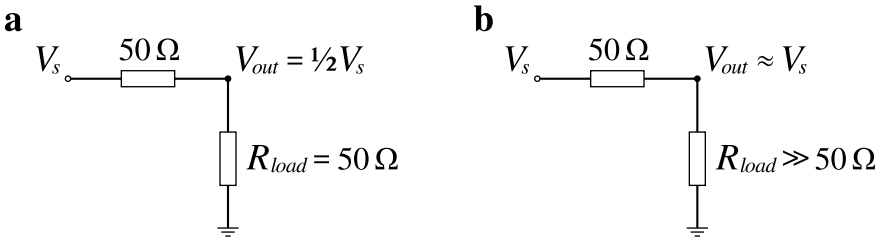


Figure C.4: Assumptions about load impedance. (a) Matched 50 Ω load. (b) High-impedance load

C.5 Code listing

```
#!/usr/bin/perl
use POSIX qw(ceil);

#  Copyright (C) 2012 Brian Standley
#
#  This program is free software: you can redistribute it and/or modify it
#  under the terms of the GNU General Public License as published by the
#  Free Software Foundation, either version 3 of the License, or (at your
#  option) any later version.
#
#  This program is distributed in the hope that it will be useful, but
#  WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
#  Public License for more details.
#
#  You should have received a copy of the GNU General Public License along
#  with this program. If not, see <http://www.gnu.org/licenses/>.

# awcgen v1.1t

$infilename  = shift @ARGV;
$outfilename = shift @ARGV;
$flags       = shift @ARGV;

$test_mode = ($flags =~ /t/);
$N_request = ($flags =~ /m/) ? 16000 : 1600;
$gain      = ($flags =~ /f/) ? 1.0   : 0.5;

if ($infilename eq "--help" || $infilename eq "-h")
{
   print "\nUsage: awcgen [input file] [output file] [flags]\n\n";
   print "  Flags:  t  \'Test mode\'    Produce two-column output for easy" .
         " plotting.\n";
   print "          m  \'More points\'  Attempt to use ~16000 points instead" .
         " of ~1600.\n";
   print "          f  \'Full scale\'   Produce a full-size waveform when" .
         " driving a\n                             50 Ohm load.\n\n";
   exit 0;
}

!($infilename =~ $outfilename)  || die "Input and output files must be" .
                                       " different.\n";
open(INFILE,  "< $infilename")  || die "Cannot open file: $infilename.\n";
open(OUTFILE, "> $outfilename") || die "Cannot open file: $outfilename.\n";

sub ReadLine
{
   my $line  = shift(@_);

   if    ($line =~ /#/)         { return (0, 0, 1);    }
   elsif ($line =~ /(.+) (.+)/) { return ($1, $2, 0);  }
   else                         { die "Bad syntax.\n"; }
}

$t_total = 0.0;
while ($line = <INFILE>)
```

```
{
   ($dt, $v, $comment) = ReadLine($line);
   if (!$comment) { $t_total += $dt; }
}

sub IsUgly
{
   my ($num, $decimal_places) = @_;

   sprintf("%e", $num) =~ /(.+)e(.+)/;
   my $scaled_mantissa  = $1 * (10.0**$decimal_places);
   return ceil($scaled_mantissa) - $scaled_mantissa > 1e-9;
}

$freq_request = $N_request/$t_total;
$div_request = 40e6/$freq_request;
$div_max = 2**34 - 1;
for ($div = ceil($div_request); IsUgly(40e6/$div, 2); $div++)
{
   if ($div > $div_max) { die "Pulse too long to handle. Try the 'm' option" .
                              " if you aren't already.\n"; }
}
$freq = 40e6/$div;
$N = int($t_total*$freq + 0.5) + 1;

printf "\n             Request       Actual\n";
printf "             ------------  ------------\n";
printf "Divider:     %-12.3f  %1.0f\n", $div_request,  $div;
printf "Frequency:   %-12.3e  %1.3e\n", $freq_request, $freq;
printf "N_points:    %-12d  %d\n\n",    $N_request,    $N;
printf "Total time:  %1.3e\n",          $t_total;
printf "Output:      %s-commpatible\n", $test_mode ? "Plot":"AWC";

if ($N < 8) { die "Pulse too short to handle.\n"; }

if (!$test_mode)
{
   printf OUTFILE "%d\n", $N;
   printf OUTFILE "%0.6e\n", $freq;
   printf OUTFILE "1\n";
   printf OUTFILE "1000.000000\n";
}

sub PrintPoint
{
   my ($t, $v) = @_;

   if ($test_mode) { printf OUTFILE "%1.6e\t%0.6f\n", $t, $v; }
   else            { printf OUTFILE "%0.6f\n", $v;            }
}

($t, $v_x) = (0.0, 0.0);
PrintPoint($t, $v_x);

seek(INFILE, 0, 0);
while ($line = <INFILE>)
{
   ($dt, $v_f, $comment) = ReadLine($line);
```

```
   if (!$comment)
   {
      $v_f *= $gain;

      if ($dt < 1e-9) { $v_x = $v_f; }
      else
      {
         $slope = ($v_f - $v_x)/$dt;
         for ($i = 0; $i < $dt*$freq; $i++)
         {
            $t += 1.0/$freq;
            $v_x += $slope/$freq;
            PrintPoint($t, $v_x);
         }
      }
   }
}

close(INFILE);
close(OUTFILE);

exit 0;
```

Appendix D

Lateral transport measurements of reduced graphite oxide

D.1 Introduction

Graphite oxide (GO) is of interest to nanoscience both as an insulator [73] and as a precursor to a graphene [33]. In the latter case, GO's is dispersed in water [31] (taking advantage of its hydrophilicity) and then deposited onto a substrate. The GO flakes may then be reduced by chemical [33] or heat treatment [84]. Graphene prepared this way is sometimes called reduced graphite oxide (rGO), and exhibits some of the properties of pristine graphene [76].

D.2 Thermal reduction of graphite oxide

Heat treatment as a method of reducing graphite oxide was demonstrated by fabricating two terminal GO devices using the standard techniques described in chapter 2. The Hummers method [32] was used to prepare the starting material, which was then deposited onto a Si/SiO_2 substrate and annealed at 420 °C for 60 seconds in H_2/Ar atmosphere.[1] From that starting point, the conductance of three individual devices was measured using a probe station before and after additional annealing. The corresponding lateral conductivity is plotted as a function of the cumulative annealing time in Figure D.1. Resistance per square before and after the additional annealing is shown in Table D.1. Two of the three devices showed at least an order of magnitude increase in conductivity while the

[1] The sample was quickly inserted into a 1-in-diameter quartz tube furnace and quickly removed after the specified time interval. The tube was continuously flushed with 0.4–2.0 slm H_2 and 0.4 slm Ar. (1 slm = 1 L/min at standard temperature and pressure.)

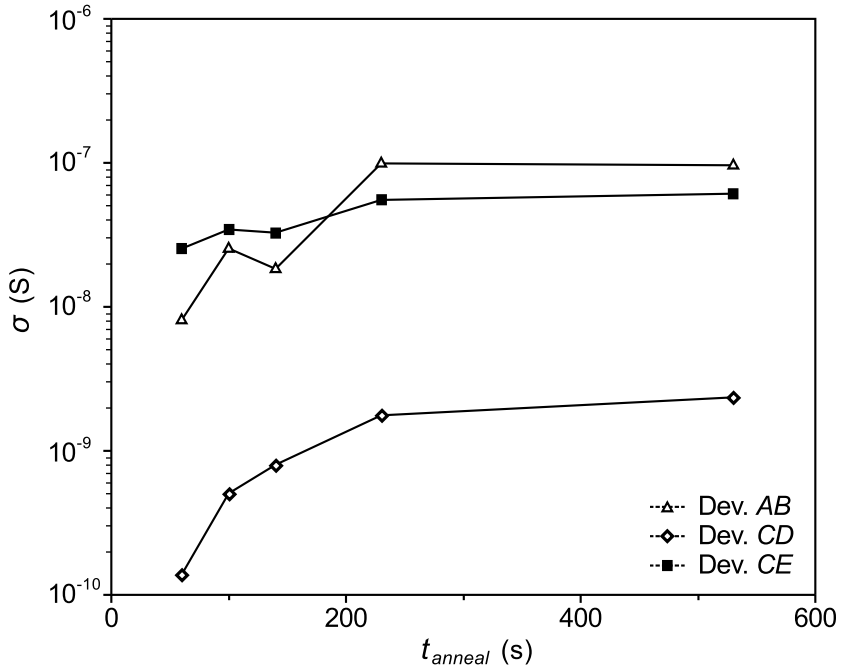


Figure D.1: Lateral conductivity of graphite oxide vs. cumulative annealing time for three rGO devices

Table D.1: Resistance per square for three rGO devices before and after annealing

Dev.	Before	After	Improvement
AB	123 MΩ	10.4 MΩ	11.8 ×
CD	7.23 GΩ	425 MΩ	17.0 ×
CE	39.4 MΩ	16.4 MΩ	2.4 ×

third changed only slightly. Given the nearly insulating nature of unreduced graphite oxide, it seems likely that at much larger change in conductivity occurred during the first 60 seconds of annealing. Although this experiment used a higher temperature than the typical ∼170 °C polymethylmethacrylate (PMMA) prebaking step, it does illustrate GO's sensitivity to heat.

D.3 Electron transport in reduced graphite oxide

The rGO devices described above were subjected to electron transport measurements after 140 s of accumulated annealing time. The I-V characteristic and back-gate sweep, shown for device *CE* in Figure D.2, are similar to those reported by Gilje et al. [33] and Gómez-Navarro et al. [78], which describes the structure of rGO as "nanometer-sized graphitic domains separated by defect clusters, which results in hopping conduction as the dominant charge-transport mechanism."

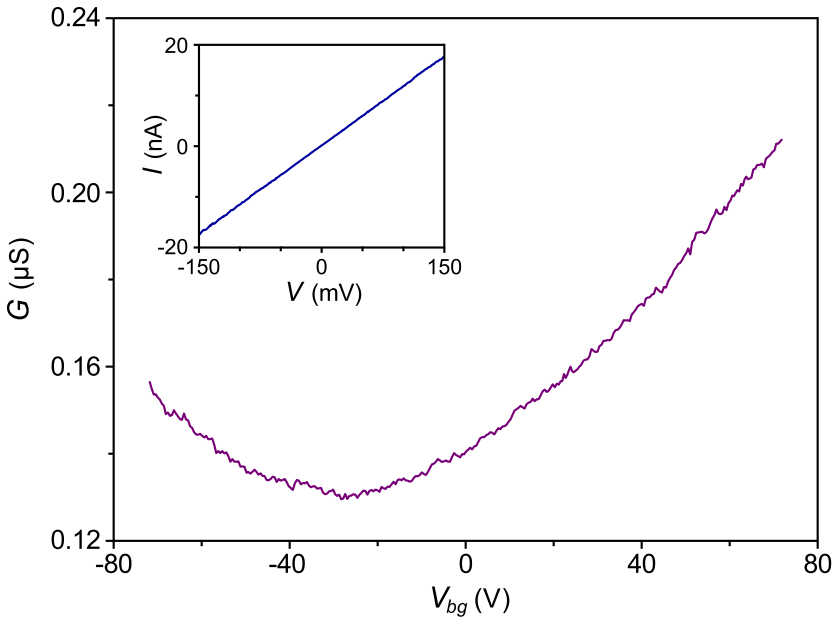


Figure D.2: Conductance vs. back-gate voltage for a rGO transistor (device *CE*). Inset: I-V characteristic at $V_{bg} = 0$ V

Appendix E

Scripting and customizing *Mezurit 2*

E.1 Introduction

Mezurit 2 can be customized in many ways, even without modifying the source code, by editing the per-user and system-wide ancillary files listed in Table E.1 or writing scripts (see sections E.3 and E.4 for examples) to control it through its *terminal* tool. The system-wide files are included with the installation package and the per-user files are created automatically the first time the program runs. (If a per-user file already exists, it will not be overwritten so as to preserve the user's modifications.) The files "compute.py" and "terminal.py" must be edited manually, while (per-user) file "default.mcf" may be updated to match the current configuration using a menu option and file "last.mcf" is automatically written immediately before the program closes.

Table E.1: *Mezurit 2* ancillary files. The second column specifies the typical location when installed on a GNU/Linux-based system, with '~' being the user's home directory.

Filename	Typical location	Description
PER-USER		
default.mcf	~/.config/mezurit2/	Default configuration
last.mcf	~/.config/mezurit2/	Last configuration
compute.py	~/.config/mezurit2/	Custom channel functions and trigger commands
terminal.py	~/.config/mezurit2/	Custom terminal commands
history	~/.config/mezurit2/	*Terminal* command history
SYSTEM-WIDE		
default.mcf	/usr/share/mezurit2/	Default configuration
mezurit2compute.py	/usr/lib/mezurit2/	Built-in channel functions and trigger commands
mezurit2control.py	/usr/lib/mezurit2/	Built-in terminal commands
terminal_startup.py	/usr/share/mezurit2/	*Terminal* configuration

E.2 Startup and shutdown processes

The sequence in which the ancillary files are verified and loaded is listed below. This information is relevant when adding material to compute.py or terminal.py and when migrating older MCF (*Mezurit 2* configuration format) files to a newer version of the program.

1. Verify per-user files.
   - ↳ Verify that the user configuration directory exists. If not, create an empty directory.
   - ↳ Verify that file "compute.py" exists. If not, create a blank file.
   - ↳ Verify that file "terminal.py" exists. If not, create a blank file.
2. Initialize the *Python* environment needed to evaluate channel and trigger functions.
   - ↳ Import the functions defined in file "mezurit2compute.py".[1]
     - ↳ Execute file "compute.py", which may extend the built-in functions.
3. Load initial configuration.
   - ↳ Load internal defaults.[2]
   - ↳ If it exists, load (per-user) file "default.mcf" and then skip to step 4. Otherwise:
     - ↳ Load (system-wide) file "default.mcf".
     - ↳ Save to (per-user) file "default.mcf" for future use.
4. Launch the separate *terminal* process running the *Python* interpreter.
   - ↳ Execute file "terminal_startup.py".
     - ↳ Load the user's command history into memory.
     - ↳ Import the functions defined in file "mezurit2control.py".[3]
     - ↳ Execute file "terminal.py", which may extend the built-in functions.
   - ↳ Process commands until shudown.
   - ↳ Save to file "history".
5. Process for input via the GUI and process commands sent from the *terminal* until shutdown.
6. Save configuration to file "last.mcf".

[1] This module, in turn, loads an internal module (_mezurit2compute) enabling low-level communication with the main process.

[2] Every MCF node has an internal default, which ensures complete initialization in the event that the default MCF file is incomplete.

[3] This module, in turn, loads another module (_mezurit2control) implementing socket-based communication with the main process.

E.3 Example script: mega1.py

The first example script "mega1.py" implements a very simple megasweep. A bias voltage V_b (X_1) is swept up and down at each gate voltage V_g (X_2) from −8 V to 8 V in 0.125 V increments.[4] Data from "up" and "down" sweeps are appended to separate files at the end of each cycle. The initial setup is left to the user, including turning on recording and configuring the sweep parameters for X_1. At completion V_g will be left at 8 V and V_b will remain at its lower limit.

E.3.1 Code listing

```
bias_ch = 1
for Vg in make_range(-8, 8, 128) :
   set_dac(2, Vg)

   clear_buffer(0, 0)
   sweep_up(bias_ch)
   catch_sweep('max_posthold', bias_ch)
   save_data('DevX_RunY_up.dat')

   clear_buffer(0, 0)
   sweep_down(bias_ch)
   catch_sweep('min_posthold', bias_ch)
   save_data('DevX_RunY_down.dat')
```

E.4 Example script: mega2.py

The second example script "mega2.py" takes over much of the responsibility for the megasweep from the user. Parameters for the sweep, such as destination filename, step sizes, sweep limits, etc., appear at the top of the script. These values are then programmed into the *sweep* tool automatically before the beginning of the loop. In addition, the megasweep itself is non-rectangular so as to track a hypothetical mechanical resonance which shifts in frequency f (X_1) as a quadratic function of gate voltage V_g (X_2) [123, 124]. The collection of measured points is shown in Figure E.1. At the end, recording is turned off and V_g is returned to zero. Two key features of the script are highlighted:

- The step size in f (ΔX_1) is set to 0.1 MHz while the limits of the sweep are rounded to the nearest 0.1 MHz. This ensures that every measured point falls on a regular grid for easy plotting.
- A settling time is included to allow external instruments to readjust after each frequency step. Equation 5.4 is then used to find the optimal sweep rate given the fixed step size.

[4] Note that there will actually be 129 sweeps in each direction to cover both endpoints of the range.

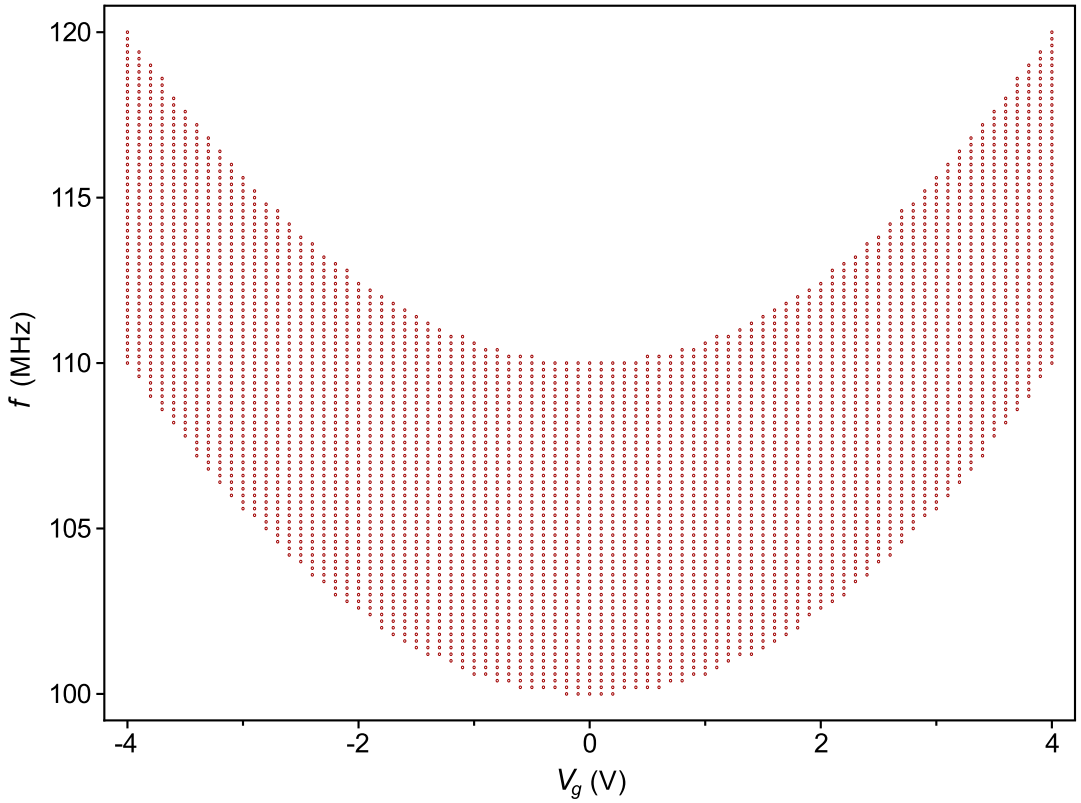


Figure E.1: Measurement points for hypothetical non-rectangular f vs. V_g megasweep. For clarity, only every second point is shown.

E.4.1 Code listing

```
basefile = 'DevX_RunY'
freq_ch = 1
f_step = 0.1
t_set = 0.4
f_min = lambda Vg : round(100 + Vg**2 / 1.6, 1)
f_max = lambda Vg : round(110 + Vg**2 / 1.6, 1)
gate_ch = 2
gate_range = make_range(-4, 4, 80)

def set_pair (node, down_val, up_val) :
   (p, i) = (get_panel(), get_sweep_id(freq_ch))
   set_var('panel{0}_sweep{1}_{2}_down={3}'.format(p, i, node, down_val))
   set_var('panel{0}_sweep{1}_{2}_up={3}'.format(p, i, node, up_val))

if (arg(clear_buffer(1, 0), 0) == '1') :
   set_recording(1)

   set_pair('step', f_step, f_step)
   set_pair('rate', f_step / t_set * 0.8, f_step / t_set * 0.8)
   set_pair('blackout', 80, 80)
   set_pair('endstop', 1, 1)
```

```
for Vg in gate_range :

   set_pair('scaled', f_min(Vg), f_max(Vg))
   set_pair('scaled', f_min(Vg), f_max(Vg))  # repeat to be sure
   set_dac(freq_ch, f_min(Vg))
   set_dac(gate_ch, Vg)

   clear_buffer(0, 0)
   sweep_up(freq_ch)
   catch_sweep('max_posthold', freq_ch)
   save_data(basefile + '_up.dat')

   clear_buffer(0, 0)
   sweep_down(freq_ch)
   catch_sweep('min_posthold', freq_ch)
   save_data(basefile + '_down.dat')

set_recording(False)
set_dac(gate_ch, 0)
```

Bibliography

[1] K. S. Novoselov, A. K. Geim, S. V. Morozov, D. Jiang, Y. Zhang, S. V. Dubonos, I. V. Grigorieva, and A. A. Firsov. Electric Field Effect in Atomically Thin Carbon Films. *Science*, 306(5696):666–669, 2004.

[2] J. Boardman, I. E. S. Edwards, N. G. L. Hammond, and E. Sollberger, editors. *The Cambridge Ancient History*, volume 3. Cambridge University Press, second edition, 1982.

[3] P. R. Wallace. The Band Theory of Graphite. *Physical Review*, 71:622–634, 1947.

[4] H. P. Boehm, A. Clauss, G. O. Fischer, and U. Hofmann. Das adsorptionsverhalten sehr dünner Kohlenstoff-Folien. *Zeitschrift für anorganische und allgemeine Chemie*, 316(3–4):119–127, 1962.

[5] A. K. Geim and A. H. MacDonald. Graphene: Exploring Carbon Flatland. *Physics Today*, 60(8):35–41, 2007.

[6] Free Software Foundation, Inc. GNU General Public License (Version 3), 2007. Available: `http://www.gnu.org/licenses/gpl.html`.

[7] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim. The electronic properties of graphene. *Reviews of Modern Physics*, 81:109–162, 2009.

[8] J. Heo. *Probing Electronic Properties of Carbon Nanotubes*. PhD thesis, California Institute of Technology, 2008.

[9] S. Das Sarma, S. Adam, E. H. Hwang, and E. Rossi. Electronic transport in two-dimensional graphene. *Reviews of Modern Physics*, 83:407–470, 2011.

[10] A. K. Geim and K. S. Novoselov. The rise of graphene. *Nature Materials*, 6(3):183–191, 2007.

[11] J. Martin, N. Akerman, G. Ulbricht, T. Lohmann, J. H. Smet, K. von Klitzing, and A. Yacoby. Observation of electron-hole puddles in graphene using a scanning single-electron transistor. *Nature Physics*, 4(2):144–148, 2008.

[12] T. Fang, A. Konar, H. Xing, and D. Jena. Carrier statistics and quantum capacitance of graphene sheets and ribbons. *Applied Physics Letters*, 91(9):092109, 2007.

[13] N. Agraït, A. L. Yeyati, and J. M. van Ruitenbeek. Quantum properties of atomic-sized conductors. *Physics Reports*, 377(2–3):81–279, 2003.

[14] B. J. van Wees, H. van Houten, C. W. J. Beenakker, J. G. Williamson, L. P. Kouwenhoven, D. van der Marel, and C. T. Foxon. Quantized Conductance of Point Contacts in a Two-Dimensional Electron Gas. *Physical Review Letters*, 60:848–850, 1988.

[15] A. G. Rinzler, J. H. Hafner, P. Nikolaev, P. Nordlander, D. T. Colbert, R. E. Smalley, L. Lou, S. G. Kim, and D. Tománek. Unraveling Nanotubes: Field Emission from an Atomic Wire. *Science*, 269(5230):1550–1553, 1995.

[16] S. Datta. *Electronic Transport in Mesoscopic Systems*. Cambridge University Press, 1997.

[17] N. D. Lang and Ph. Avouris. Oscillatory Conductance of Carbon-Atom Wires. *Physical Review Letters*, 81:3515–3518, 1998.

[18] Y. Zhang, J. P. Small, W. V. Pontius, and P. Kim. Fabrication and electric-field-dependent transport measurements of mesoscopic graphite devices. *Applied Physics Letters*, 86(7):073104, 2005.

[19] S. Liu and C. R. Loper Jr. The formation of kish graphite. *Carbon*, 29:547–555, 1991.

[20] J. S. Bunch, S. S. Verbridge, J. S. Alden, A. M. van der Zande, J. M. Parpia, H. G. Craighead, and P. L. McEuen. Impermeable Atomic Membranes from Graphene Sheets. *Nano Letters*, 8(8):2458–2462, 2008.

[21] S. P. Koenig, N. G. Boddeti, M. L. Dunn, and J. S. Bunch. Ultrastrong adhesion of graphene membranes. *Nature Nanotechnology*, 6(9):543–546, 2011.

[22] K. S. Novoselov, D. Jiang, F. Schedin, T. J. Booth, V. V. Khotkevich, S. V. Morozov, and A. K. Geim. Two-dimensional atomic crystals. *Proceedings of the National Academy of Sciences of the United States of America*, 102(30):10451–10453, 2005.

[23] J.-E. Song, T.-Y. Ko, and S.-M. Ryu. Raman Spectroscopy Study of Annealing-Induced Effects on Graphene Prepared by Micromechanical Exfoliation. *Bulletin of the Korean Chemical Society*, 31(9):2679–2682, 2010.

[24] A. N. Sidorov, M. M. Yazdanpanah, R. Jalilian, P. J. Ouseph, R. W. Cohn, and G. U. Sumanasekera. Electrostatic deposition of graphene. *Nanotechnology*, 18(13):135301, 2007.

[25] S. F. McKay. Expansion of Annealed Pyrolytic Graphite. *Journal of Applied Physics*, 35(6):1992–1993, 1964.

[26] R. R. Nair, P. Blake, A. N. Grigorenko, K. S. Novoselov, T. J. Booth, T. Stauber, N. M. R. Peres, and A. K. Geim. Fine Structure Constant Defines Visual Transparency of Graphene. *Science*, 320(5881):1308, 2008.

[27] P. Blake, E. W. Hill, A. H. Castro Neto, K. S. Novoselov, D. Jiang, R. Yang, T. J. Booth, and A. K. Geim. Making graphene visible. *Applied Physics Letters*, 91(6):063124, 2007.

[28] A. Reina, X. Jia, J. Ho, D. Nezich, H. Son, V. Bulovic, M. S. Dresselhaus, and J. Kong. Large Area, Few-Layer Graphene Films on Arbitrary Substrates by Chemical Vapor Deposition. *Nano Letters*, 9(1):30–35, 2009.

[29] K. S. Kim, Y. Zhao, H. Jang, S. Y. Lee, J. M. Kim, K. S. Kim, J.-H. Ahn, P. Kim, J.-Y. Choi, and B. H. Hong. Large-scale pattern growth of graphene films for stretchable transparent electrodes. *Nature*, 457(7230):706–710, 2009.

[30] X. Li, W. Cai, J. An, S. Kim, J. Nah, D. Yang, R. Piner, A. Velamakanni, I. Jung, E. Tutuc, S. K. Banerjee, L. Colombo, and R. S. Ruoff. Large-Area Synthesis of High-Quality and Uniform Graphene Films on Copper Foils. *Science*, 324(5932):1312–1314, 2009.

[31] S. Stankovich, D. A. Dikin, G. H. B. Dommett, K. M. Kohlhaas, E. J. Zimney, E. A. Stach, R. D. Piner, S.-B. T. Nguyen, and R. S. Ruoff. Graphene-based composite materials. *Nature*, 442(7100):282–286, 2006.

[32] W. S. Hummers and R. E. Offeman. Preparation of Graphitic Oxide. *Journal of the American Chemical Society*, 80(6):1339–1339, 1958.

[33] S. Gilje, S. Han, M. Wang, K. L. Wang, and R. B. Kaner. A Chemical Route to Graphene for Device Applications. *Nano Letters*, 7(11):3394–3398, 2007.

[34] J. S. Bunch, A. M. van der Zande, S. S. Verbridge, I. W. Frank, D. M. Tanenbaum, J. M. Parpia, H. G. Craighead, and P. L. McEuen. Electromechanical Resonators from Graphene Sheets. *Science*, 315(5811):490–493, 2007.

[35] J. C. Nabity. Nanometer Pattern Generation System (Version 9.0), 2002. Website: `http://www.jcnabity.com/`.

[36] J. Moser, A. Verdaguer, D. Jimenez, A. Barreiro, and A. Bachtold. The environment of graphene probed by electrostatic force microscopy. *Applied Physics Letters*, 92(12):123507, 2008.

[37] DL Instruments LLC. Model 1211 Current Preamplifier Datasheet, 2008. Available: `http://www.dlinstruments.com/products/`.

[38] C. R. Dean, A. F. Young, I. Meric, C. Lee, L. Wang, S. Sorgenfrei, K. Watanabe, T. Taniguchi, P. Kim, K. L. Shepard, and J. Hone. Boron nitride substrates for high-quality graphene electronics. *Nature Nanotechnology*, 5(10):722–726, 2010.

[39] Y. Zhang, Y.-W. Tan, H. L. Stormer, and P. Kim. Experimental observation of the quantum Hall effect and Berry's phase in graphene. *Nature*, 438(7065):201–204, 2005.

[40] C. Berger, Z. Song, X. Li, X. Wu, N. Brown, C. Naud, D. Mayou, T. Li, J. Hass, A. N. Marchenkov, E. H. Conrad, P. N. First, and W. A. de Heer. Electronic Confinement and Coherence in Patterned Epitaxial Graphene. *Science*, 312(5777):1191–1196, 2006.

[41] J.-H. Chen, C. Jang, S. Xiao, M. Ishigami, and M. S. Fuhrer. Intrinsic and extrinsic performance limits of graphene devices on SiO_2. *Nature Nanotechnology*, 3(4):206–209, 2008.

[42] F. Miao, S. Wijeratne, Y. Zhang, U. C. Coskun, W. Bao, and C. N. Lau. Phase-Coherent Transport in Graphene Quantum Billiards. *Science*, 317(5844):1530–1533, 2007.

[43] K. S. Novoselov, A. K. Geim, S. V. Morozov, D. Jiang, M. I. Katsnelson, I. V. Grigorieva, S. V. Dubonos, and A. A. Firsov. Two-dimensional gas of massless Dirac fermions in graphene. *Nature*, 438(7065):197–200, 2005.

[44] X. Du, I. Skachko, A. Barker, and E. Y. Andrei. Approaching ballistic transport in suspended graphene. *Nature Nanotechnology*, 3(8):491–495, 2008.

[45] A. A. Balandin, S. Ghosh, W. Bao, I. Calizo, D. Teweldebrhan, F. Miao, and C. N. Lau. Superior Thermal Conductivity of Single-Layer Graphene. *Nano Letters*, 8(3):902–907, 2008.

[46] J. Moser, A. Barreiro, and A. Bachtold. Current-induced cleaning of graphene. *Applied Physics Letters*, 91(16):163513, 2007.

[47] M. Y. Han, B. Özyilmaz, Y. Zhang, and P. Kim. Energy Band-Gap Engineering of Graphene Nanoribbons. *Physical Review Letters*, 98:206805, 2007.

[48] Z. Chen, Y.-M. Lin, M. J. Rooks, and Ph. Avouris. Graphene nano-ribbon electronics. *Physica E*, 40:228–232, 2007.

[49] X. Li, X. Wang, L. Zhang, S. Lee, and H. Dai. Chemically Derived, Ultrasmooth Graphene Nanoribbon Semiconductors. *Science*, 319(5867):1229–1232, 2008.

[50] T. J. Echtermeyer, M. C. Lemme, M. Baus, B. N. Szafranek, A. K. Geim, and H. Kurz. Nonvolatile Switching in Graphene Field-Effect Devices. *IEEE Electron Device Letters*, 29(8):952–954, 2008.

[51] H. Park, A. K. L. Lim, A. P. Alivisatos, J. Park, and P. L. McEuen. Fabrication of metallic electrodes with nanometer separation by electromigration. *Applied Physics Letters*, 75(2):301–303, 1999.

[52] S. I. Khondaker and Z. Yao. Fabrication of nanometer-spaced electrodes using gold nanoparticles. *Applied Physics Letters*, 81(24):4613–4615, 2002.

[53] K. I. Bolotin, K. J. Sikes, Z. Jiang, M. Klima, G. Fudenberg, J. Hone, P. Kim, and H. L. Stormer. Ultrahigh electron mobility in suspended graphene. *Solid State Communications*, 146(9–10):351–355, 2008.

[54] C. P. Collier, E. W. Wong, M. Belohradsk, F. M. Raymo, J. F. Stoddart, P. J. Kuekes, R. S. Williams, and J. R. Heath. Electronically Configurable Molecular-Based Logic Gates. *Science*, 285(5426):391–394, 1999.

[55] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 453(7191):80–83, 2008.

[56] K. Terabe, T. Hasegawa, T. Nakayama, and M. Aono. Quantized conductance atomic switch. *Nature*, 433(7021):47–50, 2005.

[57] R. H. M. Smit, Y. Noat, C. Untiedt, N. D. Lang, M. C. van Hemert, and J. M. van Ruitenbeek. Measurement of the conductance of a hydrogen molecule. *Nature*, 419(6910):906–909, 2002.

[58] Y. H. Lee, S. G. Kim, and D. Tománek. Field-induced unraveling of carbon nanotubes. *Chemical Physics Letters*, 265(6):667–672, 1997.

[59] K. Raghavachari and J. S. Binkley. Structure, stability, and fragmentation of small carbon clusters. *Journal of Chemical Physics*, 87(4):2191–2197, 1987.

[60] N. D. Lang and Ph. Avouris. Carbon-Atom Wires: Charge-Transfer Doping, Voltage Drop, and the Effect of Distortions. *Physical Review Letters*, 84:358–361, 2000.

[61] B. Standley, W. Bao, H. Zhang, J. Bruck, C. N. Lau, and M. Bockrath. Graphene-Based Atomic-Scale Switches. *Nano Letters*, 8(10):3345–3349, 2008.

[62] H. Zhang, W. Bao, Z. Zhao, J.-W. Huang, B. Standley, G. Liu, F. Wang, P. Kratz, L. Jing, M. Bockrath, and C. N. Lau. Visualizing Electrical Breakdown and ON/OFF States in Electrically Switchable Suspended Graphene Break Junctions. *Nano Letters*, 12:1772–1775, 2012.

[63] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck. Rank Modulation for Flash Memories. *IEEE Transactions on Information Theory*, 55(6):2659–2673, 2009.

[64] C.H. Cheng, P.C. Chen, Y.H. Wu, F.S. Yeh, and A. Chin. Long-Endurance Nanocrystal TIO_2 Resistive Memory Using a TaON Buffer Layer. *IEEE Electron Device Letters*, 32(12):1749–1751, 2011.

[65] H. Park, J. Park, A. K. L. Lim, E. H. Anderson, A. P. Alivisatos, and P. L. McEuen. Nanomechanical oscillations in a single-C_{60} transistor. *Nature*, 407(6800):57–60, 2000.

[66] P. Vettiger, G. Cross, M. Despont, U. Drechsler, U. Durig, B. Gotsmann, W. Haberle, M. A. Lantz, H. E. Rothuizen, R. Stutz, and G. K. Binnig. The "Millipede"—Nanotechnology entering data storage. *IEEE Transactions on Nanotechnology*, 1(1):39–55, 2002.

[67] J. R. Williams, L. DiCarlo, and C. M. Marcus. Quantum Hall Effect in a Gate-Controlled p-n Junction of Graphene. *Science*, 317(5838):638–641, 2007.

[68] S. Kim, J. Nah, I. Jo, D. Shahrjerdi, L. Colombo, Z. Yao, E. Tutuc, and S. K. Banerjee. Realization of a high mobility dual-gated graphene field-effect transistor with AL_2O_3 dielectric. *Applied Physics Letters*, 94(6):062107, 2009.

[69] I. Meric, M. Y. Han, A. F. Young, B. Ozyilmaz, P. Kim, and K. L. Shepard. Current saturation in zero-bandgap, top-gated graphene field-effect transistors. *Nature Nanotechnology*, 3(11):654–659, 2008.

[70] D. B. Farmer, H.-Y. Chiu, Y.-M. Lin, K. A. Jenkins, F. Xia, and Ph. Avouris. Utilization of a Buffered Dielectric to Achieve High Field-Effect Carrier Mobility in Graphene Transistors. *Nano Letters*, 9(12):4474–4478, 2009.

[71] K. Zou, X. Hong, D. Keefer, and J. Zhu. Deposition of High-Quality HfO_2 on Graphene and the Effect of Remote Oxide Phonon Scattering. *Physical Review Letters*, 105:126601, 2010.

[72] X. Hong, A. Posadas, K. Zou, C. H. Ahn, and J. Zhu. High-Mobility Few-Layer Graphene Field Effect Transistors Fabricated on Epitaxial Ferroelectric Gate Oxides. *Physical Review Letters*, 102:136808, 2009.

[73] W.-Y. Fu, L. Liu, W.-L. Wang, M.-H. Wu, Z. Xu, X.-D. Bai, and E.-G. Wang. Carbon nanotube transistors with graphene oxide films as gate dielectrics. *SCIENCE CHINA Physics, Mechanics & Astronomy*, 53:828–833, 2010.

[74] A. F. Young, C. R. Dean, I. Meric, S. Sorgenfrei, H. Ren, K. Watanabe, T. Taniguchi, J. Hone, K. L. Shepard, and P. Kim. Electronic compressibility of gapped bilayer graphene. *ArXiv:1004.5556v2*, pages 1–4, 2010.

[75] I. Meric, C. Dean, A. Young, J. Hone, P. Kim, and K. L. Shepard. Graphene field-effect transistors based on boron nitride gate dielectrics. *IEEE IEDM Technical Digest*, pages 23.2.1–23.2.4, 2010.

[76] X. Wu, M. Sprinkle, X. Li, F. Ming, C. Berger, and W. A. de Heer. Epitaxial-Graphene/Graphene-Oxide Junction: An Essential Step towards Epitaxial Graphene Electronics. *Physical Review Letters*, 101:026801, 2008.

[77] K. Kumar, S. Strauf, and E. H. Yang. A Systematic Study of Graphite Local Oxidation Lithography Parameters Using an Atomic Force Microscope. *Nanoscience and Nanotechnology Letters*, 2(2):185–188, 2010.

[78] C. Gómez-Navarro, R. T. Weitz, A. M. Bittner, M. Scolari, A. Mews, M. Burghard, and K. Kern. Electronic Transport Properties of Individual Chemically Reduced Graphene Oxide Sheets. *Nano Letters*, 7(11):3499–3503, 2007.

[79] Y. Martin, D. W. Abraham, and H. K. Wickramasinghe. High-resolution capacitance measurement and potentiometry by force microscopy. *Applied Physics Letters*, 52(13):1103–1105, 1988.

[80] C. Schönenberger and S. F. Alvarado. Observation of single charge carriers by force microscopy. *Physical Review Letters*, 65:3162–3164, 1990.

[81] J. E. Stern, B. D. Terris, H. J. Mamin, and D. Rugar. Deposition and imaging of localized charge on insulator surfaces using a force microscope. *Applied Physics Letters*, 53(26):2717–2719, 1988.

[82] M. Bockrath, N. Markovic, A. Shepard, M. Tinkham, L. Gurevich, L. P. Kouwenhoven, M. W. Wu, and L. L. Sohn. Scanned Conductance Microscopy of Carbon Nanotubes and λ-DNA. *Nano Letters*, 2(3):187–190, 2002.

[83] C. Staii, A. T. Johnson, and N. J. Pinto. Quantitative Analysis of Scanning Conductance Microscopy. *Nano Letters*, 4(5):859–862, 2004.

[84] Z. Wei, D. Wang, S. Kim, S.-Y. Kim, Y. Hu, M. K. Yakes, A. R. Laracuente, Z. Dai, S. R. Marder, C. Berger, W. P. King, W. A. de Heer, P. E. Sheehan, and E. Riedo. Nanoscale Tunable Reduction of Graphene Oxide for Graphene Electronics. *Science*, 328(5984):1373–1376, 2010.

[85] S. Roddaro, P. Pingue, V. Piazza, V. Pellegrini, and F. Beltram. The Optical Visibility of Graphene: interference Colors of Ultrathin Graphite on SiO_2. *Nano Letters*, 7(9):2707–2710, 2007.

[86] A. I. Kingon, J.-P. Maria, and S. K. Streiffer. Alternative dielectrics to silicon dioxide for memory and logic devices. *Nature*, 406(6799):1032–1038, 2000.

[87] D. C. Marcano, D. V. Kosynkin, J. M. Berlin, A. Sinitskii, Z. Sun, A. Slesarev, L. B. Alemany, W. Lu, and J. M. Tour. Improved Synthesis of Graphene Oxide. *ACS Nano*, 4(8):4806–4814, 2010.

[88] R. J. W. E. Lahaye, H. K. Jeong, C. Y. Park, and Y. H. Lee. Density functional theory study of graphite oxide for different oxidation levels. *Physical Review B*, 79:125435, 2009.

[89] C. M. Osburn and D. W. Ormond. Dielectric Breakdown in Silicon Dioxide Films on Silicon. *Journal of the Electrochemical Society*, 119(5):597–603, 1972.

[90] B. Özyilmaz, P. Jarillo-Herrero, D. Efetov, D. A. Abanin, L. S. Levitov, and P. Kim. Electronic Transport and Quantum Hall Effect in Bipolar Graphene p-n-p Junctions. *Physical Review Letters*, 99:166804, 2007.

[91] J. Velasco Jr., G. Liu, W. Bao, and C. N. Lau. Electrical transport in high-quality graphene pnp junctions. *New Journal of Physics*, 11(9):095008, 2009.

[92] B. Huard, J. A. Sulpizio, N. Stander, K. Todd, B. Yang, and D. Goldhaber-Gordon. Transport Measurements Across a Tunable Potential Barrier in Graphene. *Physical Review Letters*, 98:236803, 2007.

[93] J. B. Oostinga, H. B. Heersche, X. Liu, A. F. Morpurgo, and L. M. K. Vandersypen. Gate-induced insulating state in bilayer graphene devices. *Nature Materials*, 7(2):151–157, 2008.

[94] G. Liu, J. Velasco Jr., W. Bao, and C. N. Lau. Fabrication of graphene p-n-p junctions with contactless top gates. *Applied Physics Letters*, 92(20):203103, 2008.

[95] R. V. Gorbachev, A. S. Mayorov, A. K. Savchenko, D. W. Horsell, and F. Guinea. Conductance of p-n-p Graphene Structures with "Air-Bridge" Top Gates. *Nano Letters*, 8(7):1995–1999, 2008.

[96] T. Lohmann, K. von Klitzing, and J. H. Smet. Four-Terminal Magneto-Transport in Graphene p-n Junctions Created by Spatially Selective Doping. *Nano Letters*, 9(5):1973–1979, 2009.

[97] A. F. Young and P. Kim. Quantum interference and Klein tunnelling in graphene heterojunctions. *Nature Physics*, 5(3):222–226, 2009.

[98] Y.-M. Lin, A. Valdes-Garcia, S.-J. Han, D. B. Farmer, I. Meric, Y. Sun, Y. Wu, C. Dimitrakopoulos, A. Grill, Ph. Avouris, and K. A. Jenkins. Wafer-Scale Graphene Integrated Circuit. *Science*, 332(6035):1294–1297, 2011.

[99] Y.-W. Tan, Y. Zhang, H. L. Stormer, and P. Kim. Temperature dependent electron transport in graphene. *The European Physical Journal—Special Topics*, 148:15–18, 2007.

[100] S. V. Morozov, K. S. Novoselov, M. I. Katsnelson, F. Schedin, D. C. Elias, J. A. Jaszczak, and A. K. Geim. Giant Intrinsic Carrier Mobilities in Graphene and Its Bilayer. *Physical Review Letters*, 100:016602, 2008.

[101] M. Ishigami, J. H. Chen, W. G. Cullen, M. S. Fuhrer, and E. D. Williams. Atomic Structure of Graphene on SiO_2. *Nano Letters*, 7(6):1643–1648, 2007.

[102] D.W. Horsell, A.K. Savchenko, F.V. Tikhonenko, K. Kechedzhi, I.V. Lerner, and V.I. Fal'ko. Mesoscopic conductance fluctuations in graphene. *Solid State Communications*, 149(27–28):1041–1045, 2009.

[103] B. Standley, A. Mendez, E. Schmidgall, and M. Bockrath. Graphene-Graphite Oxide Field-Effect Transistors. *Nano Letters*, 12(3):1165–1169, 2012.

[104] A. J. M. Giesbers, U. Zeitler, S. Neubeck, F. Freitag, K. S. Novoselov, and J. C. Maan. Nanolithography and manipulation of graphene using an atomic force microscope. *Solid State Communications*, 147(9–10):366–369, 2008.

[105] L. Weng, L. Zhang, Y. P. Chen, and L. P. Rokhinson. Atomic force microscope local oxidation nanolithography of graphene. *Applied Physics Letters*, 93(9):093107, 2008.

[106] S. Masubuchi, M. Ono, K. Yoshida, K. Hirakawa, and T. Machida. Fabrication of graphene nanoribbon by local anodic oxidation lithography using atomic force microscope. *Applied Physics Letters*, 94(8):082107, 2009.

[107] R. K. Puddy, P. H. Scard, D. Tyndall, M. R. Connolly, C. G. Smith, G. A. C. Jones, A. Lombardo, A. C. Ferrari, and M. R. Buitelaar. Atomic force microscope nanolithography of graphene: Cuts, pseudocuts, and tip current measurements. *Applied Physics Letters*, 98(13):133120, 2011.

[108] V. Ananthanarayanan and W. Thies. Biocoder: A programming language for standardizing and automating biology protocols. *Journal of Biological Engineering*, 4(1):13, 2010.

[109] International Technology Roadmap for Semiconductors, 2011.
Available: `http://www.itrs.net/`.

[110] Community cleverness required. *Nature*, 455(7209):1–1, 2008.

[111] National Instruments Corporation. LabVIEW (Version 2011), 2011.

[112] Mathworks, Inc. Instrument Control Toolbox (Version 3.1), 2012.

[113] Keithly Instruments, Inc. Automated Characterization Suite Basic Edition (Version 901-01), 2011.

[114] V. A. Sazonova. *A Tunable Carbon Nanotube Resonator*. PhD thesis, Cornell University, 2006.

[115] H. W. Ch. Postma. DAQ: Data Aquisition (Version 12.02), 2012.
Available: `http://www.csun.edu/~hpostma/`.

[116] D. C. Ince, L. Hatton, and J. Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485–488, 2012.

[117] C. Tennis. Data acquisition with comedi. *Linux Journal*, 2004(124), 2004.

[118] E. A. Lee. The Problem with Threads. *Computer*, 39:33–42, 2006.

[119] B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. Prentice Hall, 2nd edition, 1988.

[120] RibbonSoft GmbH. QCAD (Version 2.2.2.0), 2008.
Website: `http://www.ribbonsoft.com/en/qcad`.

[121] T. Taychatanapat and P. Jarillo-Herrero. Electronic Transport in Dual-Gated Bilayer Graphene at Large Displacement Fields. *Physical Review Letters*, 105:166601, 2010.

[122] Stanford Research Systems, Inc. Arbitrary Waveform Composer (Version 1.0.5), 2006.
Available: `http://www.thinksrs.com/downloads/soft.htm`.

[123] V. Sazonova, Y. Yaish, H. Ustunel, D. Roundy, T. A. Arias, and P. L. McEuen. A tunable carbon nanotube electromechanical oscillator. *Nature*, 431(7006):284–287, 2004.

[124] H.-Y. Chiu, P. Hung, H. W. Ch. Postma, and M. Bockrath. Atomic-Scale Mass Sensing Using Carbon Nanotube Resonators. *Nano Letters*, 8(12):4342–4346, 2008.